ISSN 0868-6157

КОМПЬЮТЕР ПРЕСС



1'92


С НОВЫМ ГОДОМ!

Совместное советско-американское предприятие «СОВАМИНКО»

КОМПЬЮТЕР ПРЕСС



КОМПЬЮТЕРПРЕСС

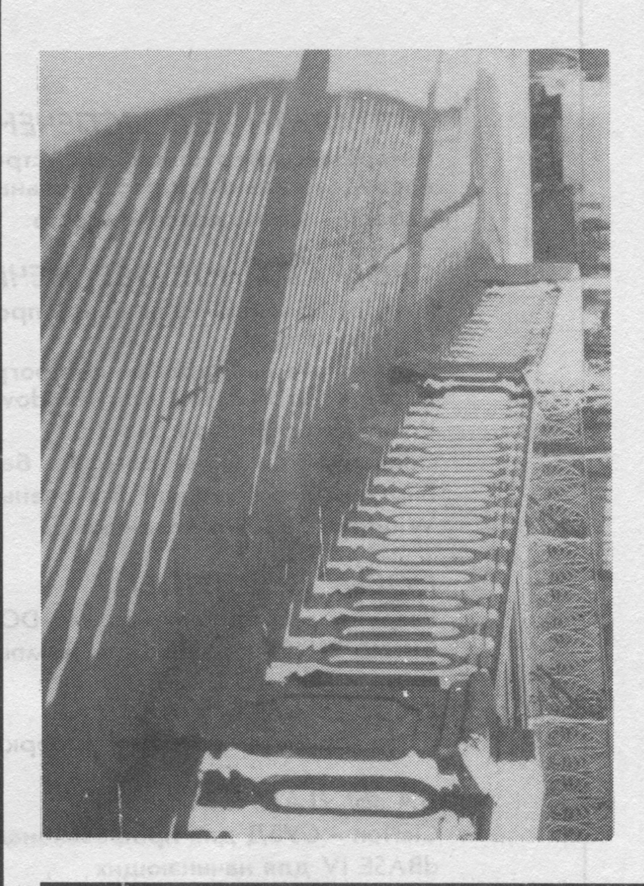
Издается с 1989 года

Выходит 12 раз в год.

1'92 (25)

Тексты проверены системой "ОРФО"

Адрес редакции:
113093, г.Москва, аб.ящик 37
Факс: 200-22-89
Телефоны для справок:
491-01-53, 420-83-80.
E-mail: postmaster@Computerpress.msk.su



Сдано в набор 12.12.91. Подписано к печати 16.12.91.
Формат 84x108/16. Печать офсетная. Бумага офсетная.
Усл.печ.л. 8,4+0,42 (обл.) Тираж 100000 экз.
(1 завод--70000). Заказ 2419. Цена 3 р. 80 к.

Оригинал-макет подготовлен агентством "КомпьютерПресс".

Отпечатано в полиграфической фирме
«Красный пролетарий» РГИИЦ «Республика»
103473, Москва, Краснопролетарская, 16.

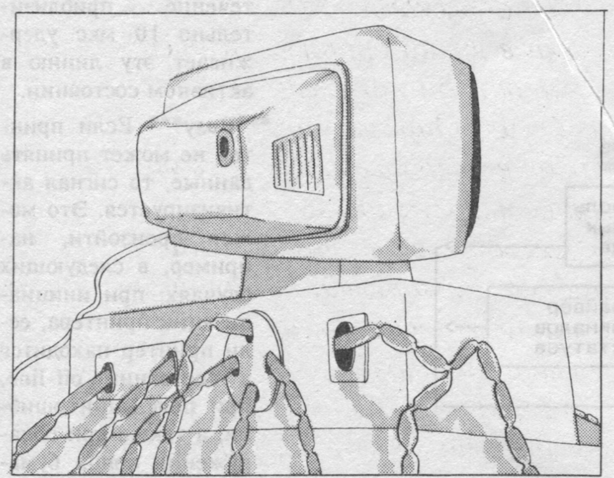
Дорогой читатель, ты держишь в руках юбилейный — двадцать пятый — номер нашего журнала. С чем тебя, а также себя мы и поздравляем. Когда мы начинали, а было это два с небольшим года назад, все было не так: и время было не то, и законы, и компьютеры были не те, да и страна была не та. Еще не было ни малых предприятий, ни цветных лазерных принтеров, ни бирж, ни ноутбуков, ни ГКЧП. Зато были КПСС, КГБ и хлеб по 20 копеек. И даже время от времени колбаса. Что лучше — ощущать себя относительно сытым или относительно свободным — каждый решает сам. Впрочем, в нашей чудной стране каждое принятое решение в итоге оказывается неверным...

Когда мы взялись за издание этого журнала, то, как выяснилось, даже отдаленно не представляли тех трудностей, с которыми столкнемся, но, как бы то ни было, журнал выходит более-менее регулярно (хотя такой регулярности не пожелаем своим читателям), бумажно-полиграфические проблемы относительно решены (не спорим, бывает и лучше), и главное для нас сегодня — найти решение вечного вопроса: как угодить максимальному числу читателей. Это всегда являлось нашей целью. Ведь чем мы отличаемся от других компьютерных журналов, выходящих в нашей стране? В них в основном заложена одна схема: перевод статей из весьма уважаемых западных журналов. Но что годится для западного читателя, зачастую не подходит нашему. Конечно, интересен совет, что для решения такой-то проблемы достаточно заменить 386/20 на 486/50, но у нас, пожалуй, многих больше интересует, что можно выжать из старенькой XT´ишки. Поэтому мы стараемся, начав и продолжая линию как можно более объективных обзоров событий компьютерного мира и сообщая о всех достойных внимания новостях, давать информацию, ориентированную исключительно на конкретные нужды нашего небогатого, но крайне любознательного и смекалистого читателя. Всем угодить невозможно, это понятно: начинающим надо одно, просто пользователям другое, крутым программистам — третье, электронщикам — четвертое и т.д. Статья, с упоением прочитанная одним, вызовет скуку у другого — это неизбежно, как невозможно совершенство. В книге Лоуренса Питера “Принцип Питера“ приведен отрывок из армейских инструкций: “Колышки

для палаток... должны быть окрашены в оранжевый цвет. Яркая окраска — лучшее средство, чтобы быстро отыскать колышки... При пользовании колышками, окрашенными в яркий оранжевый цвет, их следут вбивать в землю глубоко, так чтобы они не были видны“. Конечно, можно согласиться со словами Грушо Маркса, что “военная мудрость есть понятие, содержащее противоречие в самом себе“, но в данном случае все верно: либо колышки видны неприятелю, либо они не видны никому. Сегодня мы видим решение этой проблемы в ориентировании на письма читателей, на принцип “о чем просят, о том споем“. Так что пишите письма и, не стесняясь, рисуйте вопросы, как поет Булат Шалвович: “Ничего, что мы чужие. Вы рисуйте! Я потом, что не понятно, объясню“. Более того, написавшие самые толковые письма, возможно, получат некоторое удовлетворение в виде пачки дензнаков — в этом номере мы сообщаем о новом конкурсе на лучшую публикацию года, правда, итоги прошлогоднего конкурса удастся подвести, видимо, не ранее третьего номера.

Дорогой читатель! Позволь с некоторым опозданием поздравить тебя с Новым годом и пожелать тебе в первую очередь бодрости и оптимизма. Без оптимизма в наше нелегкое время просто не прожить, а бодрость нужна, чтобы справляться с отдельными временными трудностями. Ну, и конечно — удачи! А мы сделаем все, чтобы помочь тебе хоть в чем-то — хотя бы в решении компьютерных проблем.

Б.Молчанов, главный редактор



ИНТЕРФЕЙСЫ ПЕРИФЕРИЙНЫХ УСТРОЙСТВ — ПОСЛЕДОВАТЕЛЬНЫЙ И ПАРАЛЛЕЛЬНЫЙ

Подсоединение периферийных устройств, таких как мышка или принтер, к персональному компьютеру производится через так называемые устройства сопряжения (адаптеры). Обычно адаптеры выполнены в виде отдельных плат ввода-вывода (Input-Output Card, I/O Card), вставляемых в разъемы расширения на системной плате.

Вообще говоря, взаимодействие периферийного устройства с адаптером происходит с одним (возможно одним из двух) стандартным интерфейсом, определяющим, в частности, тип и "род" соединителя, уровни и длительности электрических сигналов, протоколы обмена. К одной плате ввода-вывода может подключаться несколько устройств, если, конечно, на ней конструктивно размещено несколько адаптеров устройств, которые в случае персональных компьютеров часто называют портами ввода-вывода. Порты бывают последовательные, т.е. информационные биты передаются последовательно один за другим, и параллельные, когда несколько бит данных передаются одновременно. Для подключения джойстика служит специальный аналоговый игровой адаптер — Game Adapter. Обычно он располагается на плате ввода-вывода вместе с параллельным и последовательными портами. Плату со всеми этими портами называют, как правило, многофункциональной платой ввода-вывода (Multi I/O Card). Параллельные порты используются обычно для подключения принтера (в ряде случаев плоттера или сканера). Интерфейс параллельного порта выполнен в соответствии с интерфейсом Centronics (получившим свое название по имени американской фирмы — производителя принтеров, предложившей в свое время собственный интерфейс для принтера). Сегодня практически все принтеры, предназначенные для персоналок, используют интерфейс Centronics, так что при подключении принтера к компьютеру проблем не возникает.

CENTRONICS: байт за байтом

Интерфейс Centronics использует электрические сигналы TTL-уровня (+5 В и 0 В) и еще не так давно был чаще всего конструктивно выполнен на нескольких TTL-микросхемах. Эти микросхемы служат для декодирования адреса, промежуточного хранения и инвертирования отдельных сигналов (рис. 1). В последнее время широкое распространение получили параллельные адаптеры, в которых практически все функции отдельных TTL-микросхем объединены в одной БИС 82С11, вы-

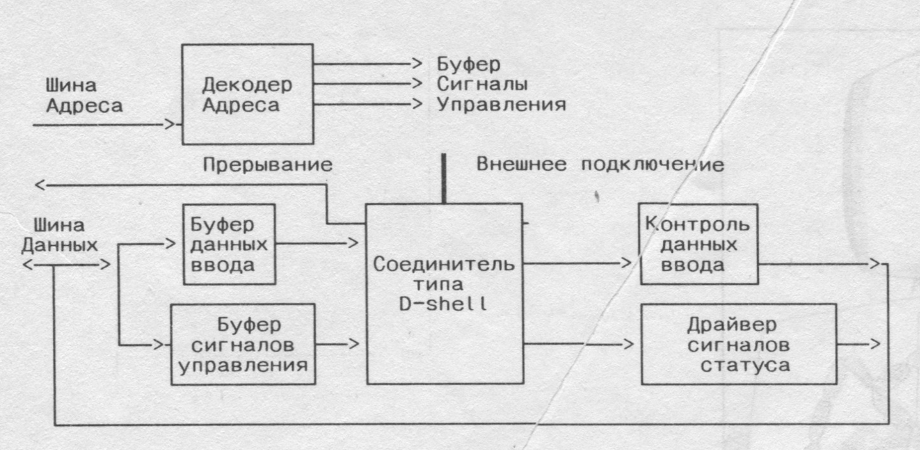


Рис. 1.

течение приблизительно 10 мкс удерживает эту линию в активном состоянии.

Busy Если принтер не может принять данные, то сигнал активизируется. Это может произойти, например, в следующих случаях: при инициализации принтера, если принтер находится в состоянии off-line, при появлении ошибки, а также при обнаружении конца бумаги.

полненной по КМОП-технологии (уровни сигналов по-прежнему TTL). Обычно эта микросхема расположена на плате в специальной панельке — chip socket — и не зря. Электрические наводки в кабеле (не говоря уже о разных "землях" принтера и компьютера) достаточно часто выводят эту микросхему из строя (КМОП — капризная вещь!). В таких случаях можно лишь пожалеть о "старой доброй" TTL-технике.

Для того чтобы избежать ошибок и потери информации при передаче данных с TTL-уровнями, максимальная длина кабеля для принтера не должна быть больше двух-трех метров. Обычно для сигналов данных и управления в кабеле используются витые пары, а все проводники заключены в общий экран. С помощью специальных дополнительных устройств (усилителей сигналов) длина кабеля может быть без особых проблем увеличена до десяти, а то и до двадцати метров. Основной задачей усилителей при этом является сохранение временных соотношений сигналов, их уровней и фронтов. Подсоединение кабеля к адаптеру (для пользователя — к компьютеру) производится через 25-контактный разъем типа D—shell, распределение сигналов по контактам которого приведено на рис.2. Со стороны принтера используется специальный 36-контактный разъем типа Centronics; распределение сигналов по контактам показано на рис.3. Надо сказать, что для простой передачи данных требуются не все сигналы, определенные стандартом Centronics. Для того чтобы обеспечить функционирование интерфейса, достаточно использовать только 8 бит данных (D0 — D7), стробсигнал данных (Data Strobe) и сигнал занятости принтера (Busy). Пара слов о сигналах интерфейса Centronics, используемых для работы с принтерами.

Data Strobe Когда компьютер посылает данные на принтер, он в течение 5 мкс должен активировать этот сигнал. Этим принтеру сообщается о том, что данные на соответствующих шинах готовы.

Data По этим 8 сигнальным линиям данные передаются от компьютера к принтеру. После установления сигнала Data Strobe принтер читает эту информацию.

Acknowledge Если принтер принял выставленные компьютером данные, то в подтверждение он в

Paper out Этот сигнал сообщает компьютеру о том, что закончилась бумага. Если вставить в принтер новый лист, сигнал дезактивируется.

Select С помощью этого сигнала принтер сообщает машине, что он выбран и активен. У многих принтеров данный сигнал имеет постоянное значение.

Auto feed Активирование этого сигнала вызывает продвижение бумаги в принтере на одну строку вперед. Как правило, переход на другую строку осуществляется программно — выводом в порт определенных символов.

Контакт	Направление	Сигнал
1	Выход	Data Strobe
2	Выход	Data 0
3	Выход	Data 1
4	Выход	Data 2
5	Выход	Data 3
6	Выход	Data 4
7	Выход	Data 5
8	Выход	Data 6
9	Выход	Data 7
10	Вход	Acknowledge
11	Вход	Busy
12	Вход	Paper Out
13	Вход	Select
14	Выход	Auto feed
15	Вход	Error
16	Выход	Init
17	Выход	Select Input
18–25	–	Ground

Рис.2.

Error Этот сигнал от принтера может быть активным в следующих ситуациях: если принтер находится в состоянии off-line, если закончилась бумага или во время печати произошла ошибка.

Init Если сигнал на этой линии будет активным в течение приблизительно 50-100 мкс, то происходит инициализация принтера — как при его включении.

Select Input Активирование и дезактивирование этого сигнала аналогично подаче управляющих кодов DC1 (Device Control 1) — выбор устройства и DC2 — отмена выбора устройства.

Контакт	Направление	Сигнал
1	Выход	Data Strobe
2	Выход	Data 0
3	Выход	Data 1
4	Выход	Data 2
5	Выход	Data 3
6	Выход	Data 4
7	Выход	Data 5
8	Выход	Data 6
9	Выход	Data 7
10	Вход	Acknowledge
11	Вход	Busy
12	Вход	Paper Out
13	Вход	Select
14	Выход	Auto feed
15	–	No connect
16	–	Gnd
17	–	Shassis Gnd
18	Выход	+5 V
19-30	–	Gnd
31	Вход	Init
32	Выход	Error
33	–	Gnd
34		Clock
35	Выход	Test
36	Выход	Select Input

Рис.3.

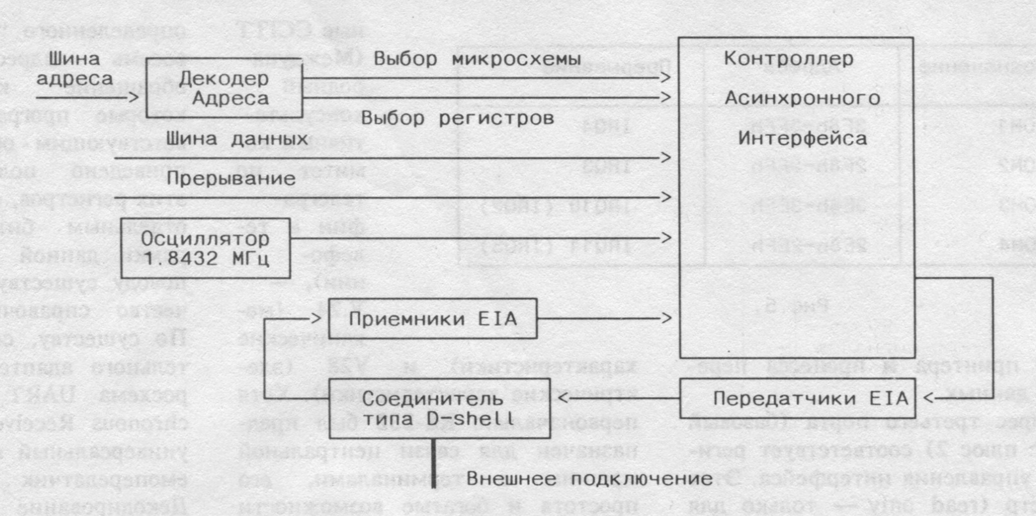


Рис.4

Ground Это сигнал "Корпус" ("Земля") для сигналов данных и управляющих сигналов.

Персональный компьютер работает максимум с тремя параллельными портами, которые в MS-DOS имеют логические имена lpt1, lpt2 и lpt3. В адресном пространстве компьютера резервируются базовые адреса этих портов: 3BCh, 378h и 278h. Первый адрес обычно используется, если принтер-порт находится, например, на плате графического адаптера Hercules или EGA. На плате Multi I/O Card адрес lpt1 378h, а lpt2 — 278h. Для принтерного порта lpt1 предусмотрено аппаратное прерывание IRQ7, а для lpt2 — IRQ5, хотя на практике они используются очень редко. Установка базовых адресов портов и возможность использования прерываний настраивается перестановкой перемычек (Jumpers) на плате, описание которых обычно приведено в технической документации на плату.

Начиная с базового адреса, каждый адаптер принтера имеет в адресном пространстве три адреса. При этом первый адрес соответствует регистру данных, посылаемых от компьютера к принтеру. В случае использования TTL-микросхем этот регистр часто бывает реализован на микросхеме 74LS374. Чтение установленных битов данных можно осуществить по тому же адресу. Физически чтение данных происходит через буфер данных, выполненный, например, на микросхеме 74LS244. Следующий адрес (базовый адрес плюс единица) позволяет читать регистр статуса адаптера (расположенный, конечно, в принтере) через буферную микросхему — часто 74LS240 (или 74LS367 — 74LS368). В регистре статуса биты 3-7 позволяют определить состояние некоторых сигналов интерфейса Centronics:

бит 3 = 0: Error
бит 4 = 1: Select
бит 5 = 1: Paper out
бит 6 = 0: Acknowledge
бит 7 = 0: Busy

Чтение регистра статуса имеет смысл при передаче данных на принтер, для определения состо-

Обозначение	Адреса	Прерывание
COM1	3F8h-3FFh	IRQ4
COM2	2F8h-2FFh	IRQ3
COM3	3E8h-3EFh	IRQ10 (IRQ2)
COM4	2E8h-2EFh	IRQ11 (IRQ5)

Рис.5.

яния принтера и процесса передачи данных.

Адрес третьего порта (базовый адрес плюс 2) соответствует регистру управления интерфейса. Этот регистр (read only — только для чтения), для которого может использоваться микросхема 74LS174, позволяет определить следующие состояния принтера:
бит 0 = 0: сигнал Data Strobe активен,
бит 1 = 0: сигнал Auto feed включен,
бит 2 = 0: инициализация принтера,
бит 3 = 1: принтер выбран,
бит 4 = 1: прерывание разрешено.

Для того чтобы работать с принтером из собственных программ, проще всего использовать специальное прерывание BIOS int 17h. Однако, в некоторых случаях может потребоваться программирование на уровне регистров, тогда информация о приведенных выше регистрах просто необходима.

Интерфейс в мир RS-232

Последовательный интерфейс может использоваться для большинства периферийных устройств: плоттер, удаленный принтер, мышь, модем, программатор ПЗУ и т.д. До настоящего времени для последовательной связи используются адаптеры с интерфейсом RS-232. Описание этого интерфейса было опубликовано Американской промышленной ассоциацией еще в 1969 году. Европейским аналогом RS-232 являются два стандарта, разработанные CCITT (Международный консультативный комитет по телеграфии и телефонии), — V.24 (механические характеристики) и V28 (электрические характеристики). Хотя первоначально RS-232 был предназначен для связи центральной машины с терминалами, его простота и богатые возможности обеспечили ему более широкое применение. В современной персоналке может быть до четырех последовательных адаптеров, имеющих логические имена соответственно COM1, COM2, COM3 и COM4. Структурная схема одного из адаптеров приведена на рис.4. Базовые адреса адаптеров и соответствующие прерывания приведены на рис.5. Следует обратить внимание на тот факт, что использование прерываний IRQ10 и IRQ11 в IBM PC/AT возможно только на дополнительном (коротком) слоте. Таким образом, применение трех или четырех последовательных портов с соответствующими прерываниями возможно только на плате ввода-вывода для PC/AT (16 бит данных). В противном случае можно задействовать только два прерывания (IRQ4 и IRQ3) или использовать, если возможно, прерывание IRQ2 или IRQ5.

В адресном пространстве портов PC последовательный адаптер занимает восемь последовательных адресов, включая базовый. Однако с помощью определенного “трюка“ через эти восемь адресов происходит обращение к 11 регистрам, которые программируются соответствующим образом. На рис.6 приведено полное обозначение этих регистров, описание же их по отдельным битам выходит за рамки данной статьи (по этому поводу существует большое количество справочной литературы). По существу, сердцем последовательного адаптера является микросхема UART (Universal Asynchronous Receiver/Transmitter) — универсальный асинхронный приемопередатчик 8250 или 16450. Декодирование адресов для этой микросхемы происходит либо на дискретных TTL-микросхемах, либо при помощи микросхемы программируемой логической матрицы (Programmable Array Logic). Поскольку стандарт передачи и приема использует высокие уровни сигналов +/-15 В или +/-12 В, то для преобразования логических уровней в интерфейсные и наоборот используются микросхемы приемников и передатчиков, обычно 1489 и 1488. Каждая из этих микросхем включает в себя четыре приемника или четыре передатчика. При передаче микросхема UART преобразует параллельный код в последовательный и передает его побитно в линию, обрамляя исходную последовательность битами старта, останова и контроля. При приеме данных UART преобразует последовательный код в параллельный (разумеется, опуская служебные символы). Непременным условием правильной передачи/приема яв-

Адрес	Обозначение
0	THR Transmitter Holding Register
0	RBR Receiver Buffer Register
0	DLL Divisor Latch LSB
1	DLM Divisor Latch MSB
1	IER Interrupt Enable Register
2	IIR Interrupt Identification Register
3	LCR Line Control Register
4	MCR Modem Control Register
5	LSR Line Status Register
6	MSR Modem Status Register

Рис.6.

ляется одинаковая скорость работы приемного и передающего UART. Основным преимуществом последовательной передачи является возможность пересылки данных на большие расстояния — как правило, не менее 30 метров. Немаловажно и то, что в простейшем случае для приема и передачи необходимы только три сигнала: TxD (Transmit Data — Передача данных), RxD (Receive Data — Прием данных) и GND (Ground — Земля).

В персональных компьютерах из 25 сигналов, предусмотренных стандартом RS-232, используются в соответствии с EIA только 9: три описанных выше и шесть, объединенных общим названием Handshake-сигналы. При использовании этого интерфейса одно из устройств выступает как DTE (Data Terminal Equipment — Оконечное устройство), а другое как DCE (Data Communication Equipment — Устройство передачи данных). Хотя, вообще говоря, различие между ними состоит только в направлении используемых сигналов. Так, если сигнал для DTE является входным, то для DCE этот же сигнал будет выходным и наоборот. На рис. 7 приведено распределение по контактам девяти используемых сигналов для 25-контактного и 9-контактного разъемов. Кстати, 9-контактный разъем для последовательного адаптера используется только на PC/AT. При обмене данными могут использоваться различные протоколы — правила обмена — от простейшего, упомянутого выше и работающего только с тремя сигналами интерфейса, до более сложных, использующих, например, пару квитирующих (handshake) сигналов RTS-CTS. Различные программы могут применять различные протоколы обмена, поэтому во избежание недоразумений лучше всегда предварительно изучить соответствующие технические описания.

Скорость последовательной передачи данных оценивается в бодах (бит полезной информации в секунду). Микросхемы UART 8250 рассчитаны на максимальную скорость 38400 бод, а 16450 — на 115200 бод. Обычно передача данных осуществляется на нескольких дискретных скоростях: 50, 75, 110, 150, 300, 600, 1200, 2400, 4800, 9600 и 19200 бод. Для IBM PC максимальная скорость обмена обычно составляет 9600, а для PS/2 — 19200 бод. При специальном программировании регистров можно достигнуть скорости обмена в 115200 бод. Различные коммерческие программы (типа LapLink) используют эту возможность. Конечно, чем выше скорость обмена, тем выше требования к используемому кабелю, чтобы избежать ошибок при приеме или передаче. Средства BIOS (например, коммуникационное прерывание int 14h) поддерживают скорости только до 9600 бод включительно. Это следует иметь в виду при написании собственных программ обмена. Тактовая частота, используемая для UART, обычно составляет 18432 кГц и стабилизирована благодаря использованию кварцевого генератора. Из этой частоты формируются уже все упомянутые частоты. Последовательный порт работает асинхронно — данные передаются без тактового сигнала. В этом случае незначительное различие скоростей приема и передачи не влияет на качество обмена. На обеих сторонах должны быть установлены программно (или с помощью переключателей) следующие параметры: скорость передачи данных, количество битов обмена (от 5 до 8), количество стоповых битов (1 или 2), бит контроля (по четности или нечетности, при 8 битах отсутствует). Передача данных начинается с изменения уровня напряжения на линии с -12 В до уровня +12 В (так называемый стартовый бит). Стоповые биты передаются уровнем напряжения -12 В. При использовании контроля по четности, соответствующий бит выбирается таким образом, что сумма битов данных и бита контроля представляет собой четное число. Аналогично выполняется контроль по нечетности.

В заключение, пожалуй, нужно

Обозначение сигнала	Контакты DB9	Контакты DB25
DCD	1	8
RxD	2	2
TxD	3	3
DTR	4	20
GND	5	7
DSR	6	6
RTS	7	4
CTS	8	5
RI	9	22

Рис. 7.

Номер контакта	Направление сигнала	Название сигнала
1	Выход	+5 В
2	Вход	Клавиша 1 джойстик А
3	Вход	Значение X джойстик А
4	–	Земля
5	–	Земля
6	Вход	Значение Y джойстик А
7	Вход	Клавиша 2 джойстик А
8	Выход	+5 В
9	Выход	+5 В
10	Вход	Клавиша 1 джойстик В
11	Вход	Значение X джойстик В
12	–	Земля
13	Вход	Значение Y джойстик В
14	Вход	Клавиша 2 джойстик В
15	Выход	

Рис. 8.

сказать несколько слов об адаптере джойстика, который, как правило, также расположен на Multi I/O Card. Распределение сигналов по контактам разъема для игрового адаптера приведено на рис.8. Одной из особенностей этого разъема является то, что к нему, в отличие, например, от разъемов последовательного или параллельного адаптера, могут подключаться два джойстика одновременно. Такое соединение выполняется обычно посредством так называемого Y-кабеля (два входа, один выход). Для тех, кто увлекается компьютерными играми, нет нужды объяснять необходимость именно двух джойстиков. Хотя, по правде говоря, на современном компьютере джойстик выглядит анахронизмом.

В адресном пространстве для Game-адаптера зарезервировано поле адресов от 200h до 207h. На самом деле для этого адаптера достаточно только одного адреса. Для PC/XT это, как правило, 200h, а для PC/AT — 201h. Этим различием можно объяснить то, что иногда текстовые или игровые программы не распознают игровой адаптер.

А.Борзенко

Китай и Иран создают новые волоконные линии

В то время, как в Россию и другие республики бывшего Советского Союза ввоз оптико-волоконных кабелей по-прежнему запрещен Западом, потому что разговоры по ним нельзя перехватить, у Китая и Ирана такой проблемы нет. Китай создает волоконную линию длиной 2.500 км, соединяющую Шанхай с Гуанчжоу. Иран закупил оптико-волоконную систему у шведской фирмы Ericsson для северо-западного района Ирана вблизи от иракской границы. Вскоре Ericsson откроет отделение в Тегеране для дальнейшего развития бизнеса. Эти новости появились как раз тогда, когда пресса сообщила, что Китай, возможно, помогал Ирану в его проекте создания атомной бомбы.

The Teleputing Hotline, November 12, 1991

Улучшения в передаче сообщений между системами не за горами

Стандарт X.400 для передачи сообщений между системами нуждается в единой адресации, и предпринимаются шаги по ее созданию. В прошлом, однако, компании действовали сами по себе, и для создания общего стандарта потребуются переговоры. British Telecom соединит между собой свои системы в Соединенном Королевстве, США и Франции таким образом, что пользователи в этих стран будут пользоваться единой схемой адресации и электронной почтой к январю 1992 года. Передача сообщений между этими службами еще потребует использования X.400 Message Switching Service, которая работает на сети BT Tymnet. Но существует и новое автоматическое программное обеспечение.

Американская Pacific Bell добавила схему адресации X.400 к своей службе передачи сообщений, чтобы все пользователи могли вместе создать базу данных удаленных адресов, которой можно пользоваться при создании списков рассылки. Information Network фирмы IBM представила EmailConnect for Trading Partners — службу, клиенты которой могут создать свои собственные системы адресации или получить помощь IBM при преобразовании адресов поставщиков и покупателей.

Наконец, и это может оказаться самым важным, Retix внесет свой вклад, начав перенос и обновление адресов сетей стандарта X.400 с помощью Directory Exchange (DX). DX мог бы стать сердцем "X.500" — нового стандарта адресации. Программное обеспечение Retix подходит к системам X.400 самых различных типов компьютеров.

The Teleputing Hotline, November 12, 1991

Электронное оформление счетов развивается: под двумя стандартами

Электронная обработка счетов по стандартам EDI набирает популярность в мире, но испозуются два разных стандарта. Европа поддерживает Edifact, а американские фирмы стоят за X.12.

British Telecom (BT) раскидывает свою службу EDI-Net по Великобритании и Европе. В 1985 году пакетная сеть Tymnet начала работу своей службы EDI. Теперь, когда владельцем сети является BT, две системы будут объединены. BT также предлагает жесткую связь с британскими компаниями и большим числом европейских узлов. Плата за подписку отсутствует.

В Москве группа развития системы Edifact под названием PEPI сообщила корреспонденту Newsbytes Кириллу Чащину, что железные дороги и пароходства переходят от бумажек к электронной обработке счетов. Чтобы ускорить этот процесс, PEPI обеспечит доступ к базе данных по документам по Edifact в Женеве и будет продавать программное обеспечение французской фирмы Transpak.

В США EDI-служба фирмы Sprint начнет пересылку счетов на факсимильные аппараты, в электронные почтовые ящики, системы BBS и по почте, в соответствии с X.12. Это делается для того, чтобы фирма могла пользоваться стандартом EDI, даже если ее деловые партнеры по-прежнему пользуются бумажной документацией.

The Teleputing Hotline, November 12, 1991

Cognito представила Radio Transmission Unit (RTU) для переносных компьютеров. RTU предоставляет пользователям переносных PC доступ к мобильным сетям передачи данных. Noel Leslie, управляющий маркетингом фирмы Cognito, хочет связаться с разработчиками по поводу создания прикладных программ. Устройства пересылают данные со скоростью 6.000 бит в секунду, сообщениям для поиска адресата в интерактивной сети требуется 7 секунд. Разница между RTU и модемом заключается в том, что RTU можно подсоединить практически к любому устройству обработки данных, включая крупные персональные компьютеры, принтеры или пакетные сети передачи данных. С момента начала работы в августе, Cognito подключила 20 компаний с примерно 200 терминалами. Теоретически существующая радиосеть на УКВ частотах, которая к следующему марту должна охватить территорию с примерно 70% населения Великобритании, может обслуживать около 70.000 пользователей.

The Teleputing Hotline, November 14, 1991



"Не забудь проверить кабели".
(7-ое правило пользователя PC)

Кабели — это не совсем просто...

Приобретая персональный компьютер, редко кто задумывается о разнообразных кабелях для подсоединения периферийных устройств, входящих обычно в комплект поставки. Вспоминают о них лишь тогда, когда приподнятое настроение, вызванное удачной покупкой, уже проходит и начинается рутинная повседневная работа. Выясняется, например, что длина кабеля для подключения плоттера чуть меньше необходимой, а принтер, который предполагалось установить у самой стены (как всегда, мало места), из-за типа соединителя на кабеле разместить, как хотелось бы, не удается. Влияние электромагнитных помех и шумов, вызванных работой различной электро- и радиоаппаратуры, часто сказывается на функционировании удаленных периферийных устройств.

Нетрудно согласиться, что проблем может возникнуть немало, хотя многие из них можно было бы предусмотреть заранее. Конечно, покупая компьютер, у, скажем так, не очень серьезной "фирмы", вы вряд ли сможете сразу приобрести именно те кабели, которые вам необходимы. Скорее всего вам смогут помочь немногочисленные официальные дистрибьюторы и производители компьютеров, появившиеся на нашем "узком" компьютерном горизонте. Для того чтобы выбирать, надо знать, как и из чего.

Давайте, особенно не углубляясь в детали, поговорим о кабелях для периферийных устройств.

Обычно на вопрос — "Какие бывают кабели?" — можно получить столь же оригинальный ответ — "Разные". (Ну как не вспомнить диалог чеховских героев: "А какое правление в Турции? — Известно какое, турецкое".) И, тем не менее, кабели действительно бывают разные, хотя, конечно, имеют и много общего. Кстати, не очень благозвучное слово "кабель" (если верить словарю) в переводе с голландского означает канат или трос. С помощью кабелей к компьютеру подключаются практически все периферийные устройства: клавиатура, монитор, мышь, модем, принтер, плоттер и т.п. Открыв крышку компьютера, можно обнаружить, что винчестер и приводы флоппи-дисков подключаются к соответствующим контроллерам также через кабели (правда, несколько иные). Для того чтобы разобраться, что же все-таки отличает и объединяет все эти кабели, рассмотрим, как устроен практически любой кабель. Это, во-первых, соединители (разъемы), находящиеся на двух концах кабеля, и, во-вторых, изолированные друг от друга проводники, соединяющие эти разъемы. Причем все проводники могут быть тем или иным образом заключены в металлическую оболочку (экран), а весь кабель может быть покрыт пластиковой защитной оболочкой. Таким образом, друг от друга кабели могут отличаться типами со-

единителей на обоих концах кабеля, количеством используемых в кабеле проводников, тем, как эти проводники связаны с контактами разных соединителей. Конечно же, кабели различаются и по длине, и еще ряду параметров, некоторые из которых мы постараемся рассмотреть в дальнейшем. Итак...

Соединители

Соединители (разъемы) бывают двух видов — розетки и вилки. Контактные выводы вилок выполнены обычно в виде небольших (возможно, позолоченных) штырьков, которые при соединении с однотипным разъемом (но уже вилкой) входят в соответствующие пазы ответных контактов. Впрочем, контакты и в розетке, и в вилке могут быть выполнены в виде плоских пружинных пластин, но тогда формы самих соединителей таковы, что для обеспечения надежного контакта соединитель-вилка входит в соединитель-розетку, например, как у разъемов типа Centronics. Отечественные электронщики обычно называют соединитель-розетку "мамой", а соединитель-вилку — "папой", что вовсе не признак их сексуальной распущенности. Дело в том, что в техническом языке их западных коллег официально используются слова "male" — для соединителей-вилок и "female" — соответственно для розеток. Таким образом, можно считать, что соединители имеют "род". Сами соединители могут быть как разборными, так и неразборными (залитые в пластиковую оболочку — полная аналогия с вилками для бытовых электроприборов). Понятно, что и то и другое исполнение имеет как преимущества, так и недостатки. Если вы не собираетесь модифицировать схему соединения кабеля, то, вероятно, вам скорее подойдет неразборный соединитель. Если в кабеле используется несколько экранов, то более удачным будет разборный соединитель в металлическом корпусе. Ну, а о ремонтопригодности кабеля можно говорить, конечно, только в одном случае. Для хорошего заземления и надежного закрепления кабеля к устройству (или компьютеру) нужны два винта по краям соединителя. Эти винты могут быть под отвертку или могут иметь специальную накатанную головку (thumbsscrews), что позволяет закручивать их руками (это действительно удобно). Соединители же типа Centronics фиксируются с помощью двух прижимных скоб. В общем случае все соединители, используемые для периферии персонального компьютера, можно подразделить на три группы. К первой, пожалуй, наиболее многочисленной, относятся соединители типа D-shell (в профиль напоминающие латинскую букву D) различного размера и с разным числом выводных контактов. Они используются в кабелях для мониторов, модемов, принтеров (со стороны компьютера), плоттеров и т.п. Соединители типа Centronics используются, как правило, в кабелях для принтеров и плоттеров с аналогичным интерфейсом (на приборной стороне, разумеется). И третий тип соединителей — трубчатый штекер, используется в кабелях для клавиатуры и для подключения мышек типа Bus Mouse.

Проводники

Каждый проводник в кабеле выполнен обычно из тонких медных проволочек (прядей), обладающих лучшими механическими свойствами, нежели один медный провод, такого же диаметра. Проводник, как правило, облужен оловом, что предохраняет его от окисления и облегчает пайку к выводному контакту соединителя. Для обозначения общего диаметра провода, используемого в кабеле, используется специальный номер толщины ("у них"), обычно это номера 26 или 24, реже 28. Причем, чем тоньше провод, тем больше номер. Использование более толстых проводов связано не только с механической прочностью, но в первую очередь, конечно, с меньшим удельным сопротивлением, что особенно важно для длинных кабелей. Каждый провод обычно изолирован от других поли-хлорвиниловой оболочкой (PolyVinil Chloride, PVC). Для удобства распайки и контроля, изоляция проводов окрашивается в различные цветовые оттенки. Для длинных кабелей особую роль играет емкость кабеля, образующаяся между проводниками, где роль диэлектрика выполняет их изоляция. Чем ниже емкость кабеля, тем более высокочастотные сигналы по нему можно передавать. Уменьшить емкость, как известно, можно, уменьшив величину диэлектрической проницаемости материала изоляции. Поэтому для "низкоемкостных" (low-cap) кабелей в качестве материала изоляции используется полиэтилен. Для этого случая также подходят более "толстые" кабели (с меньшим номером толщины, например, 24). Внешняя фторопластовая оболочка для кабелей используется в случае их применения в промышленных помещениях (фторопласт, в отличие от PVC, при горении не выделяет токсичных веществ). Правда, не все кабели имеют пластиковую защитную оболочку. У плоских ленточных кабелей, служащих, например, для соединения винчестера со своим контроллером, такой оболочки нет. Эти кабели более дешевы, но предназначены для работы только на небольших расстояниях.

Экранирование

Кроме оболочки из пластика, кабели могут иметь одну или несколько оболочек, выполненных из токопроводящего материала: медная оплетка, слои алюминиевой фольги или все вместе. Это и есть экранирование кабеля. Экран (экраны) должен быть обязательно заземлен — незаземленный экран резко увеличивает емкость кабеля, так что эффективность его использования в этом случае может быть с точностью до наоборот. Функционально экранирование решает, как

правило, две задачи. Во-первых, экран препятствует влиянию внешних электромагнитных полей на полезные сигналы, передаваемые по кабелю. Источниками этих полей могут быть различные электрические машины, лампы "дневного" света и т.п. (Electro Magnetic Interference, EMI). И, во-вторых, экранируется создаваемое в самом кабеле электромагнитное поле, которое является источником радиопомех (Radio Frequency Interference, RFI). Необходимо помнить, что медная оплетка в качестве экрана лучше работает с низкочастотным шумом, а алюминиевая фольга — с высокочастотным. Если честно, то большинство периферийных устройств, расположенных вблизи компьютера, мало чувствительны к внешним (конечно, несильным) помехам и вполне надежно выполняют свои функциональные обязанности. Это касается клавиатуры, монитора, модема, близко расположенных принтера и плоттера. Однако, если речь идет об удаленном устройстве (о том же принтере), то над экранированием стоит подумать. Хороший эффект экранирования дают витые пары проводов в кабеле, так как это уменьшает взаимное влияние сигналов друг на друга. Но даже в таких кабелях витые пары размещают иногда в отдельный экран из фольги. Следует помнить, что для больших расстояний лучше использовать кабели экранированные, с толстым сечением провода и низкой удельной емкостью. Стандартный кабель имеет удельную емкость порядка 100 пикофарад на метр, для low-cap кабеля это значение должно быть раза в три меньше.

Кабели для последовательной связи

Последовательные кабели имеют много названий (data bit-wise и т.п.) и соединяют большинство периферийных устройств компьютера: клавиатуры, модемы, мыши, принтеры и плоттеры (с последовательным интерфейсом), также они могут использоваться для связи компьютеров друг с другом (небольшие локальные сети). Каждое устройство обычно требует своего типа кабеля. Если говорить о наиболее распространенном последовательном интерфейсе RS-232C (EIA-232D), то типовым соединителем для IBM PC/XT и PS/2 является D-shell, а точнее DB-25 (25 контактов), а для IBM PC/AT используется, как правило, DB-9 (9 контактов), хотя возможен вариант с DB-25. Стандартным для "писишек" IBM является решение, когда блочным соединителем служит вилка (male), а на кабеле — розетка (female). Для персоналок других фирм это не всегда так (например, Tandy или Labtam). В обозначении кабеля типа M/M или M/F первая буква говорит о "роде" соединителя со стороны PC, вторая — со стороны устройства (male-male, male-female). Уровни электрических сигналов в кабелях для последовательной связи достаточно высокие (+12 или -12 вольт), а скорости передачи достаточно низкие (300 - 9600 бит/с, может быть, правда, и выше), поэтому экранирование обеспечивает надежную передачу данных на больших расстояниях (до 60 метров). Кстати, если вы используете принтер с последовательным интерфейсом, то неплохо бы лишний раз заглянуть в техническую документацию, так как в зависимости от фирмы-изготовителя в кабеле могут использоваться дополнительные соединения (перемычки).

Соединители в кабелях для клавиатуры — это обычно трубчатые штекеры с пятью (для PC/XT/AT) или шестью контактными выводами (для PS/2). Длина такого кабеля обычно не превышает двух-трех метров.

Кабели для параллельной связи

Практически любой персональный компьютер имеет кабель для параллельной связи. Это кабель для принтера с интерфейсом типа Centronics. Кстати, этот интерфейс используют многие планшетные плоттеры (формата A3, A4). Соединитель на этом кабеле со стороны компьютера — обычно розетка DB-25, а на стороне принтера — 36-контактный разъем-вилка типа Centronics. Эти кабели могут иметь как 18, так и 25 проводников, однако, работают они со всеми принтерами, изготовленными в США. Интерфейс Centronics использует сигналы транзисторно-транзисторной логики (TTL) с уровнями 0 или +5 вольт. Поскольку частота передаваемых сигналов может достигать десятков килогерц, длина таких кабелей обычно не превышает трех метров. Как правило, для линий данных и строб-сигнала в кабеле используются витые пары.

Кабели для мониторов различаются в зависимости от типа используемого монитора. Имеется два стандарта на соединители в таких кабелях: 15-контактный DB-shell для мониторов, использующих аналоговые сигналы (VGA, SuperVGA, 8514/A и XGA), и 9-контактный DB-shell для мониторов, использующих TTL-сигналы (MDA, CGA и EGA). Как правило, на

кабелях мониторов имеются специальные фильтры, уменьшающие высокочастотные помехи в сигналах изображения. Длина таких кабелей также не превышает двух-трех метров.

Tutti-frutti, или "всякая всячина"

Производители компьютеров обычно вместе с основным предлагают многочисленное дополнительное оборудование, среди которого не последнее место занимают и кабели специального назначения. Например, для установки принтера непосредственно у стены помещения применяется специальный кабель Right Angle (вывод проводов кабеля перпендикулярен плоскости соединителя). Для того чтобы соединить два компьютера через последовательные порты, обычный кабель для модема, конечно, не подойдет, необходим специальный "нуль-модемный" (Null Modem) кабель. Имеются различного вида удлинительные кабели, которыми можно наращивать основной. Правда, приобретая такой кабель, не следует забывать о двух вещах: во-первых, о "роде" (male-female) соединителей удлиняемого кабеля, и, во-вторых, о том, что один длинный кабель внесет меньше искажений в сигнал, нежели два кабеля, соединенных вместе. При длине кабеля более 4,5 метров экранирование просто необходимо. При работе с устройствами, позволяющими, например, использовать один принтер для нескольких компьютеров, также необходим набор расширительных кабелей. Кстати, такие устройства носят милое нашему сердцу название "коллективизаторов". Предлагаются и специальные кабели (длина которых ограничена иногда длиной используемых соединителей) для изменения "рода" кабеля; имея, например, "неправильный" кабель с соединителем-розеткой и используя специальный переходник, несложно получить необходимый соединитель-вилку. Для любителей работать с комфортом (или если системный блок загромождает рабочий стол), предусмотрен удлинительный кабель к клавиатуре длиной обычно не более трех метров. То есть продумано многое, даже то, от чего мы пока еще достаточно далеки.

Резюмируя вышеизложенное, не грех, как говорится, вспомнить старую студенческую шутку о том, что "радиоэлектроника — это наука о контактах". Очень не рекомендуется выдергивать (как правило, закрепленный винтами) кабель не за его соединитель. Маловероятно, что сорвется резьба винтов, а вот, что "отлетит" какой-нибудь проводник — это реально. Ведь если соединитель неразборный, с кабелем, по-видимому, придется расстаться. Если вы используете удаленный принтер, проложите соединительный кабель так, чтобы он хотя бы не мешался под ногами, иначе вам можно гарантировать немало "приятных" минут. Поскольку в рамках одной статьи практически невозможно изложить даже весь материал, связанный, например, с экранированием, то перед приобретением компьютера (безусловно, в комплекте с кабелями) неплохо проконсультироваться со специалистами, если вы в чем-то не вполне уверены. Поверьте, кабели — это не совсем просто...

А.Борзенко

Смерть Максвелла может замутить воды американского он-лайнового бизнеса

Смерть магната прессы Роберта Максвелл (Robert Maxwell) может привести к новой перетряске среди он-лайновых служб США. Maxwell создавал маленькую империю он-лайновых служб, включающую в себя Official Airline Guide, бывшие службы BRS и Orbit, Prentice-Hall Information, National Register и Standard Rate & Data. Максвелл был также владельцем издательства Macmillan, в которое входило Que, ведущее издательство книг по компьютерам. Целью, которую поставил себе Maxwell, было образовать единый конгломерат из этих разноообразных предприятий, предлагающий бестселлеры в он-лайновом виде и на компакт-дисках и связывающий между собой различные службы. Но работа продвинулась недалеко, а сообщения в прессе указывают на то, что вся его финансовая империя была близка к катастрофе. Компании могут быть проданы, по отдельности или все вместе. Пока неясно, какие долговые обязательства за ними остались.

The Teleputing Hotline, November 12, 1991

Владельцы BBS опасаются судебных исков

Odyssey, BBS "для взрослых", опасается судебных исков после недавней шумихи вокруг Prodigy. 29 октября Анти-Диффамационная Лига Бнай Брит заявила, что обнаружила антисемитские материалы, распространяемые по электронным бюллетеням Prodigy, и Prodigy с готовностью ужесточила цензуру — теперь запрещены оскорбления как личные, так и в адрес различных групп. Владелец Odyssey Michael Allen сказал, что весь шум был вокруг сообщений, отправленных пользователями, а система не при чем. Он озабочен возможностью того, что кто-нибудь возбудит против его бюллетеня иск. Несмотря на это, Odyssey собирается прекратить цензурировать сообщения, помещаемые в определенные открытые разделы и по спорным вопросам. Планируется также создание раздела по обсуждению абортов. Odyssey доступен через CompuServe или напрямую по 818/358-6968.

The Teleputing Hotline, November 14, 1991



Итак, после долгих сомнений вы, наконец, решились написать большую программу. Это может быть, например, СУБД или некоторая интегрированная система. Перед вами сразу встает множество проблем, и одна из них заключается в том, чтобы обеспечить возможность расширения функций вашей системы без переписывания текстов программ.

Импорт объектов из внешней программы на Turbo Pascal

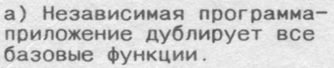
Обычно эта проблема решается при помощи макрофункций или макроязыка, позволяющего пользователю создавать свои собственные приложения. Примером программы с развитым макроязыком может служить текстовый редактор MultiEdit, в котором все операции по управлению текстом могут быть выполнены с помощью программы-макроса. Единственный недостаток этого метода заключается в сложности его реализации. Действительно, если вы решите включить в вашу программу макроязык, то вам придется писать интерпретатор и отладчик, что не так просто.

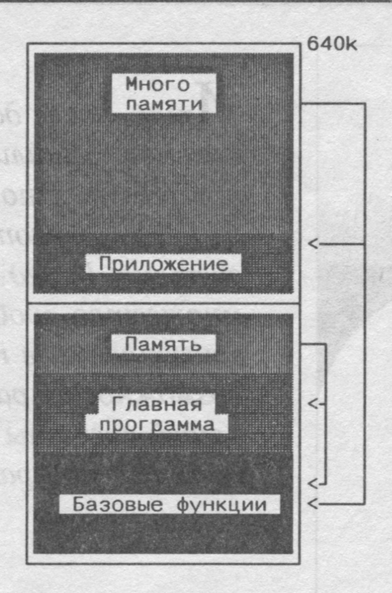
Чтобы не связываться с интерпретатором, можно написать собственную библиотеку функций, которую затем использовать в различных языках программирования. В этом случае программист сможет создавать приложения к системе, используя готовый компилятор и создавая исполняемые (.EXE) модули программ-приложений. Однако каждая такая программа должна будет содержать в себе весь набор базовых функций вашей системы (рис. 1a).

Другое решение заключается в том, что все приложения могут запускаться из главной программы и вызывать из нее (главной программы) все необходимые процедуры. Другими словами, программы-приложения будут использовать код главной программы для доступа к базовым операциям (рис. 1b). Примером такой системы может служить Microsoft Windows. Программы, написанные для этой системы, используют базовые операции для того, чтобы в графике изображать окна, меню и диалоговые панели.

Хорошая идея, но как ее реализовать? Первое, что приходит в голову, это воспользоваться системой прерываний. Главная программа перехватывает свободное прерывание, а программа-приложение затем вызывает его для доступа к базовым функциям. Однако тут мы попадаем в странное положение: обе программы




а) Независимая программа-приложение дублирует все базовые функции.



b) Программа-приложение вызывает базовые функции из главной программы.

Рис. 1. Сравнение двух способов организации доступа к базовым функциям из программы-приложения.

(главная и приложение) пишутся на языке высокого уровня с использованием объектно-ориентированного программирования, а связь между ними осуществляется примитивно, при помощи регистров и прерываний. Было бы гораздо лучше, если бы программа-приложение могла обращаться к процедурам или объектам главной программы так же, как она обращается к собственным процедурам и объектам. Для этого ей необходимо сообщить адреса всех базовых процедур, которые она будет использовать, и это можно легко сделать, используя объектно-ориентированный подход.

Давайте с вами напишем программу MAIN, которая будет содержать некий базовый объект (например, базу данных BASE), и попробуем организовать доступ к методам этого объекта из программы-приложения, которую назовем CHILD. Для начала нужно вспомнить, что при компиляции программы на Pascal все обращения к невиртуальным методам объектов заменяются на вызовы процедур с явно указанными адресами. Поэтому, если мы хотим изменять эти адреса в процессе работы программы, мы должны объявить все методы объекта BASE виртуальными. Для каждого типа объекта, имеющего виртуальные методы, Pascal создает таблицу виртуальных правил (VMT), которая содержит адреса всех виртуальных методов объекта.

Теперь нам понадобится создать два модуля (библиотеки). Первый модуль (назовем его **BASE1**) будет содержать полное описание объекта BASE со всеми методами. Второй модуль (**BASE0**) будет содержать только описание объекта BASE в разделе **interface**, а вместо процедур в разделе **implementation** будут стоять "пустые" заголовки и пустые операторы **begin end**. Первый модуль будет использован при компиляции программы MAIN, а второй — при компиляции программы CHILD. Таким образом, программа CHILD сможет использовать объект BASE, который описан в модуле BASE0. Для этого типа объекта будет создана таблица VMT, в которой будут записаны адреса "пустых" методов этого объекта. Адрес этой таблицы можно получить при помощи стандартной функции **TypeOf**. Если бы мы могли заменить этот адрес в объекте программы CHILD на адрес аналогичной таблицы из программы MAIN, то мы бы достигли желаемого результата, но, к сожалению, этого сделать нельзя, потому что объект хранит только смещение таблицы VMT. Следовательно, единственный способ переадресовать обращение к методам объекта — полностью скопировать содержимое таблицы VMT программы MAIN в аналогичную таблицу программы CHILD.

Но это еще не все. Посмотрите на рисунок 1b. Вы видите, что программа CHILD вызывает процедуры из программы MAIN. Теперь представьте, что некоторая процедура пытается отвести память под свои переменные путем вызова стандартной процедуры GetMem. Она будет вызвана из области программы MAIN и попытается выделить память в области, зарезервированной программой MAIN, которая на время выполнения программы CHILD будет уменьшена до минимума. Для того чтобы процедуры из программы MAIN могли отводить память в области, зарезервированной программой CHILD, мы должны присвоить переменным **HeapPtr** и **HeapOrg** программы MAIN значения соответствующих переменных программы CHILD.

Чтобы программа CHILD могла совершить все вышеперечисленные действия и получить доступ к методам объекта BASE, программа MAIN должна создать таблицу настройки, которая должна содержать следующие сведения:

- адрес переменной HeapPtr;
- адрес переменной HeapOrg;
- длина таблицы VMT объекта BASE;
- адрес таблицы VMT объекта BASE.

Как вычислить длину таблицы VMT? Смотрите на рисунок 2: таблица VMT содержит размер объекта (он занимает 16 бит), отрицательный размер объекта для контроля инициализации (еще столько же) и список полных адресов всех виртуальных методов. Следовательно, длина таблицы VMT вычисляется как:

```
VMT_size := 4 + SizeOf(Pointer) * Meth_Num;
```

где Meth_Num — количество виртуальных методов объекта. Количество методов не удается получить

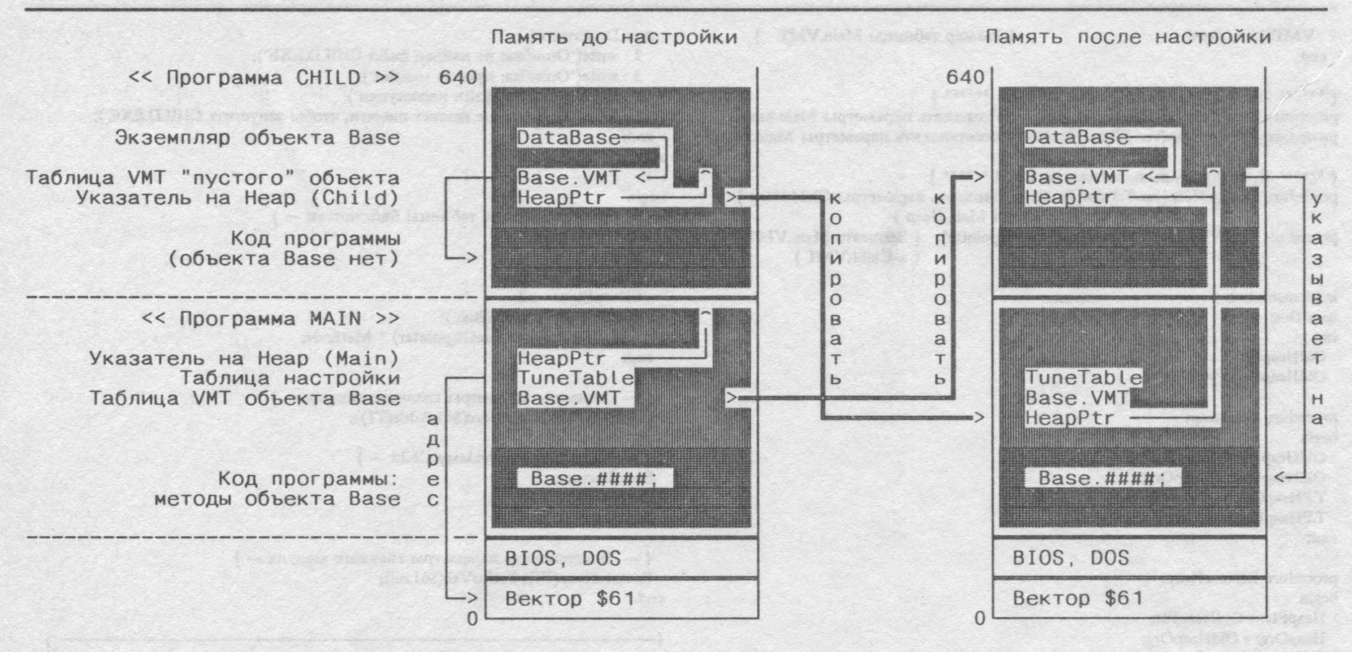


Рис. 3. Настройка программы-приложения на работу с главной программой.

стандартной функцией, поэтому в качестве этого параметра лучше использовать константу.

Размер объекта	2 байта
Отриц. размер	2 байта
Адрес 1 метода	4 байта

Рис. 2. Структура таблицы VMT.

Из таблицы настройки программа CHILD узнает все необходимое для того, чтобы настроить себя на работу с программой MAIN. Осталось сообщить ей адрес этой таблицы. Этот адрес программа MAIN поместит в один из свободных векторов прерываний (например, $61). Программа CHILD должна прочитать этот вектор и проверить его правильность, для чего в таблице настройки предусмотрен заголовок.

Сразу после запуска программа CHILD должна прочитать содержимое вектора прерывания, проверить, лежит ли по этому адресу таблица настройки и, если это так, заменить содержимое таблицы виртуальных правил (VMT) "пустого" объекта на содержимое аналогичной таблицы главной программы. После этого программа CHILD должна переслать содержимое переменных HeapPtr и HeapOrg в программу MAIN и вызвать конструктор экземпляра объекта BASE (при этом адрес таблицы VMT с новыми адресами методов будет помещен в экземпляр объекта). На этом заканчивается настройка программы CHILD (рис. 3).

После того, как программа CHILD завершит свою работу и управление вернется в программу MAIN, необходимо вернуть прежние значения переменным HeapPtr и HeapOrg (их нужно сохранить перед вызовом программы CHILD) и освободить занятый вектор прерывания.

Осталось добавить, что если вы хотите организовать импорт нескольких объектов, то вам придется реорганизовать таблицу настройки, чтобы в ней можно было передавать параметры нескольких объектов. Для этого в начале таблицы можно указать количество импортируемых объектов.

Остается открытым вопрос об использовании этого метода вместе с оверлейными модулями. Для того чтобы импортировать оверлейный объект, наверное, можно передать параметры буфера оверлеев так же, как мы передали параметры Heap.

Все, что здесь написано, я проверил на практике, написав программы MAIN и CHILD, которые успешно работают, демонстрируя правильность сделанных выводов. Ниже приведен текст обеих программ. Для удобства использования я выделил все функции настройки в отдельный модуль TUNE. Модуль BASE1 не приведен для экономии места; он отличается от модуля BASE0 тем, что в разделе **implementation** описана конкретная реализация объекта BASE.

С. Кучеров

```
{-----------------------------------------------------------------}
{                    Модуль настройки TUNE                        }
{-----------------------------------------------------------------}
unit Tune;
interface
type
  PointPtr    = ^pointer;
  TunePtr     = ^Tune_Table;
  Tune_Table  = record              { Таблица настройки объекта }
   Head      : integer;             { Заголовок всегда -1       }
   PHeapOrg  : PointPtr;            { Указатель на Main.HeapOrg }
   PHeapPtr  : PointPtr;            { Указатель на Main.HeapPtr }
   VMTPtr    : pointer;             { Указатель на Main.VMT     }
```

```
  VMTSize  : word;              { Размер таблицы Main.VMT  }
 end;

{ ****** Процедуры, вызываемые из Main ****** }
procedure SaveHeap(var T:Tune_Table);    { Сохранить параметры Main.heap }
procedure RestoreHeap(var T:Tune_Table); { Восстановить параметры Main.heap }

{ ****** Процедуры, вызываемые из Child ****** }
procedure ChangeHeap(var T:Tune_Table); { Записать параметры Child.Heap }
                                        { в Main.Heap }
procedure CopyVMT(var T:Tune_Table; Vptr:pointer);  { Записать Main.VMT }
                                                    { в Child.VMT }

implementation
uses Dos;
var
  OldHeapPtr,
  OldHeapOrg : pointer;

procedure SaveHeap;
begin
  OldHeapPtr: = HeapPtr;
  OldHeapOrg: = HeapOrg;
  T.PHeapOrg: = @HeapOrg;
  T.PHeapPtr: = @HeapPtr;
end;

procedure RestoreHeap;
begin
  HeapPtr: = OldHeapPtr;
  HeapOrg: = OldHeapOrg;
  T.PHeapOrg: = nil;
  T.PHeapPtr: = nil;
end;

procedure ChangeHeap;
begin
  with T do begin
    PHeapOrg^: = HeapOrg;
    PHeapPtr^: = HeapPtr;
  end;
end;

procedure CopyVMT;
begin
  with T do Move(VMTPtr^,Vptr^,VMTSize);
end;
end.

{----------------------------------------------------------------------}
{                       Главная программа                              }
{----------------------------------------------------------------------}
uses Crt, Dos, Tune, Base1;
{$M $400,0,0}

var TT : Tune_Table;        { таблица настройки }
  S  : string;
  i  : byte;
const
  Methods = 10;             { количество методов объекта }

procedure WriteError;
begin
  TextAttr: = $8C; gotoxy(2,4);
case DosError of
  2 : write('Ошибка: не найден файл CHILD.EXE');
  3 : write('Ошибка: путь не найден');
  5 : write('Ошибка: файл недоступен');
  8 : write('Ошибка: не хватает памяти, чтобы запустить CHILD.EXE');
 end;
end;

begin
 { --- Инициализация таблицы библиотеки --- }
 with TT do begin
   Head : = -1;
   PHeapOrg : = nil;
   PHeapPtr : = nil;
   VMTPtr : = TypeOf(Base);
   VMTSize : = 4 + SizeOf(pointer) * Methods;
 end;

 { --- Записать параметры главного модуля --- }
 SaveHeap(TT); SetIntVec($61,Addr(TT));

 { --- Запустить программу Child --- }
 SwapVectors;
 Exec('CHILD.EXE','');
 SwapVectors;

 { --- Восстановить параметры главного модуля --- }
 RestoreHeap(TT); SetIntVec($61,nil);
end.

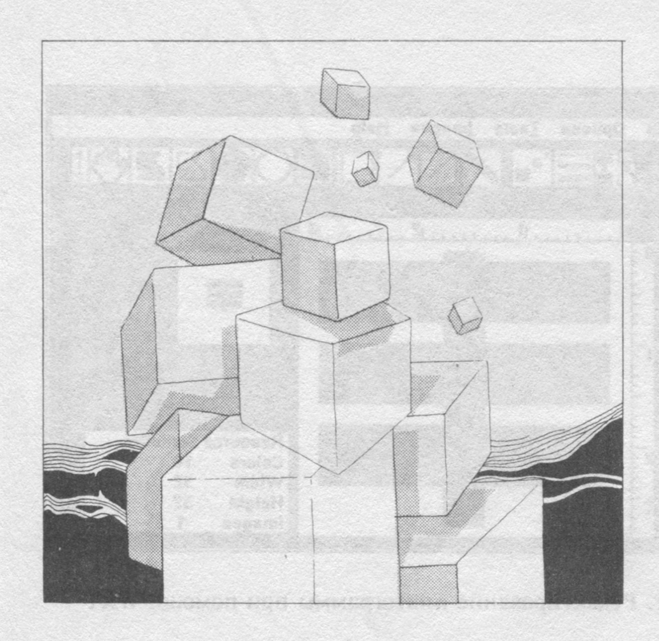
{----------------------------------------------------------------------}
{                     Программа-приложение                             }
{----------------------------------------------------------------------}
uses Crt, Dos, Tune, Base0;

var
 TT    : TunePtr;               { Указатель на таблицу настройки }
 B     : base;                  { Экземпляр импортируемого объекта }

function From_Main : boolean;
var P : pointer;

begin
 GetIntVec($61,P); TT: = P;     { Получить адрес таблицы настройки }
 From_Main: = true;
 if (TT = nil) or               { Если вектор пуст }
   (TT^.Head < > -1)            { или не указывает на таблицу, }
   then From_Main: = false      { то настройка невозможна }
 else begin
   ChangeHeap(TT^);        { переслать параметры Heap в программу MAIN }
   CopyVMT(TT^,TypeOf(Base)); { скопировать таблицу VMT из }
                      { программы MAIN в аналогичную таблицу CHILD }
 end;
end;

begin
 if not From_Main then begin
   writeln('Программа должна быть запущена из главного модуля!');
   Halt;
 end;
 B.Init;                        { Запуск конструктора объекта }
 {** Обращение к экземпляру объекта BASE.                         **}
 {** Все методы этого объекта будут вызваны из программы MAIN.    **}
end.
```



Как нельзя узнать вкус напитка по одной лишь его этикетке, едва ли можно выбрать среду программирования без практической работы в ней, только путем чтения спецификаций, пресс-релизов и журнальных обзоров. Поэтому в данной статье предлагается не обзор возможностей Turbo Pascal для Windows, а последовательное рассмотрение конкретного примера программы SysColor, которая позволит вам устанавливать системные цвета среды Microsoft Windows.

Объектно-ориентированное программирование в среде Turbo Pascal для Windows

С появлением новой версии Turbo Pascal, предназначенной для создания программ в среде Microsoft Windows, легко представить себе, как могла бы выглядеть коммерческая телереклама этого программного продукта. Например, один программист подменяет другому дистрибутив Turbo Pascal для Windows копией Microsoft SDK, а когда жертва в отчаянии говорит окружающим "сдаюсь" и получает назад свой дистрибутив, на экране появляется заставка Turbo Pascal, и счастливый обладатель пакета восклицает: "Ха-ха, я наконец-то получил самое лучшее в мире средство для программирования в Windows!".

Однако вряд ли коммерческая реклама поможет программисту решить, действительно ли выбор в качестве средства создания программ системы Turbo Pascal для Windows (TPW) является верным. В конце концов, так же, как нельзя узнать вкус напитка по одной лишь его этикетке, едва ли можно выбрать среду программирования без практической работы в ней, только путем чтения спецификаций, пресс-релизов и журнальных обзоров. Поэтому в данной статье предлагается не обзор возможностей TPW, а последовательное рассмотрение конкретного примера программы SysColor на языке Turbo Pascal для Windows, которая позволит вам устанавливать системные цвета среды Microsoft Windows.

Изучая исходный текст программы SysColor, вы познакомитесь с Объектной Библиотекой для Windows (Object Windows Library, OWL), входящей в состав TPW, и получите представление о том, как объектно-ориентированный подход позволяет снизить громоздкость, обычно присущую программам для Windows. Попутно вы получите полезную утилиту с исходным текстом, которую вы сможете свободно распространять (в том числе и по BBS) и вообще делать с ней все, что угодно.

Компиляция программы SysColor

Исходные тексты программы SysColor и файла ресурсов приводятся в конце настоящей статьи. Загрузив редактор TPW, создайте файл SYSCOLOR.PAS и наберите исходный текст программы; затем таким же образом введите текст файла ресурсов SYSCOLOR.RC.

Ввод программ вручную займет около 5 часов; если у вас есть модем, вы можете получить архив с исходными текстами на BBS. (В нашей стране исходные тексты программы SysColor можно получить, подключившись к BBS СП "Диалог". Имя BBS — "JV Dialogue WH BBS", московский телефон (095) 329-2192, область 9 (MS Windows Files). Эта BBS работает круглосуточно все дни недели. Исходные тексты про-

граммы SysColor находятся в архиве SYS-COLOR.LZH, (прим. перев.).

Раскрыв архив, вы получите три файла, нужных для компиляции программы SysColor: SYSCOLOR.PAS, SYSCOLOR.RC и SYSCOLOR.ICO.

В файле SYSCOLOR.PAS содержится исходный текст программы SysColor на Pascal. Файл SYSCOLOR.ICO содержит пиктограмму (см. рис. 1), которая отображается при установке программы SysColor в группе Microsoft Windows, при минимизации ее окна и в диалоговом окне "About...".

Файл SYSCOLOR.RC называется *файлом описания ресурсов*. Ресурсы программы для Windows — это данные, которые используются функциями Windows для отображения диалоговых окон, меню, пиктограмм и пр. Файл описания ресурсов — это текстовый файл, в котором при помощи специальных операторов описаны ресурсы Microsoft Windows. Ресурсы программы для Windows включаются компоновщиком в готовый исполняемый модуль на самом последнем этапе создания программы. Компоновщик записывает ресурсы в исполняемый файл в двоичном виде (файл *.RES, который получается из файла *.RC путем компиляции компилятором ресурсов фирмы Microsoft). Стандартные функции обслуживания диалога, известные под собирательным названием *Интерфейс Программирования Приложения* (Application Programming Interface, API), а также другие функции Microsoft Windows доступны программам на TPW из двух библиотек (TPU) — WinTypes и WinProcs.

Для того чтобы скомпилировать файл SYS-COLOR.RC, в первой строке измените путь так, чтобы он указывал на тот каталог, где у вас находится файл WINDOWS.H, после чего введите команду

RC -r syscolor

Ключ -r указывает компилятору ресурсов, что созданный файл SYSCOLOR.RES не надо вписывать в исполняемый файл SYSCOLOR.EXE (это для нас сделает Turbo Pascal).

Если вы набрали файл ресурсов вручную, то, прежде чем компилировать его, создайте пиктограмму. Для этого вызовите *Редактор Ресурсов*, входящий в состав пакета Turbo Pascal для Windows (Whitewater Resource Toolkit, WRT), вызовите редактор пиктограмм и нарисуйте пиктограмму, подобную изображенной на рис. 1. Нарисуйте три горизонтальные полосы — красную, зеленую и голубую, пересеченные изображением регулятора (или изобретите более оригинальную картинку). После этого сохраните пиктограмму в файле SYSCOLOR.ICO и скомпилируйте ресурсы.

Вообще существует два способа создания ресурсов для программы под Windows: во-первых, можно с помощью обычного текстового редактора создать файл ресурсов вручную, а потом скомпилировать его при помощи программы RC.EXE, как это показано выше.

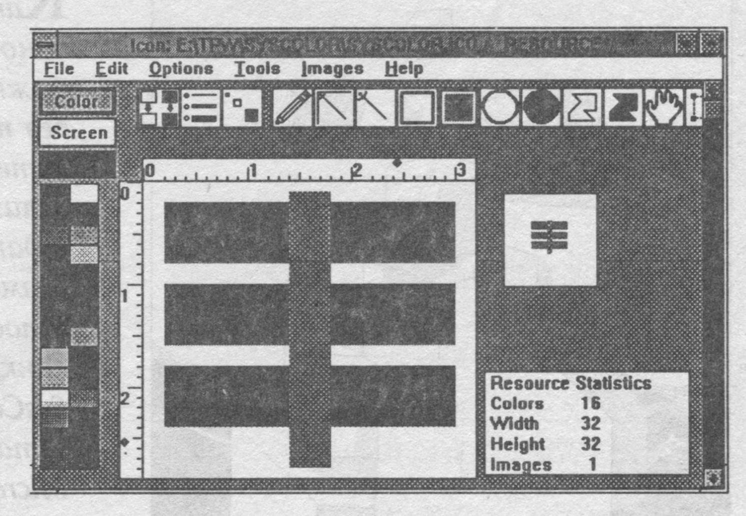


Рис. 1. Редактирование пиктограммы при помощи WRT

Заметим, что для этого нужно знать язык описания ресурсов и, кроме того, вам придется вручную вводить координаты точек на экране.

Гораздо удобнее сразу создавать файл *.RES при помощи WRT. Потом можно сохранить созданные ресурсы в файле *.RC для документации. Именно так были созданы ресурсы программы SysColor.

После создания ресурсов вызовите TPW и загрузите в него исходный текст программы SysColor — SYS-COLOR.PAS. Нажатием клавиш Ctrl-F9 вы можете прямо в Windows скомпилировать и запустить программу SysColor (компиляция без запуска программы — Alt-F9)! Как Turbo Pascal для DOS, Turbo Pascal для Windows за один прием скомпилирует программу и создаст исполняемый модуль.

Программу для Windows можно создать также и из командной строки при помощи следующей команды:

TPCW syscolor

Отметим, что если при компиляции программы в среде MS-DOS ограничение оперативной памяти в 640 Кбайт иногда бывает существенным, то компилятор TPW использует всю оперативную память ПЭВМ, поскольку является настоящим приложением Windows (а не запускает дочерний процесс MS-DOS, как можно ошибочно подумать). Поэтому, если на вашем компьютере установлена, например, расширенная память объемом 1 Мбайт, то вы можете создавать действительно гигантские программные проекты.

После того как программа SysColor будет создана, скопируйте ее в каталог Windows. Вызовите File Manager и мышью "перетащите" файл SYSCOLOR.EXE в окно любой группы. Тем самым вы добавите к сред-

ствам Windows еще одно (на этот раз свое собственное!).

Как работает программа SysColor

На рис. 2 показано окно программы SysColor. Три полосы прокрутки слева вверху позволяют пользователю изменять красную, зеленую и голубую компоненты цвета, отображаемого в большом прямоугольнике с подписью Color в центре окна:

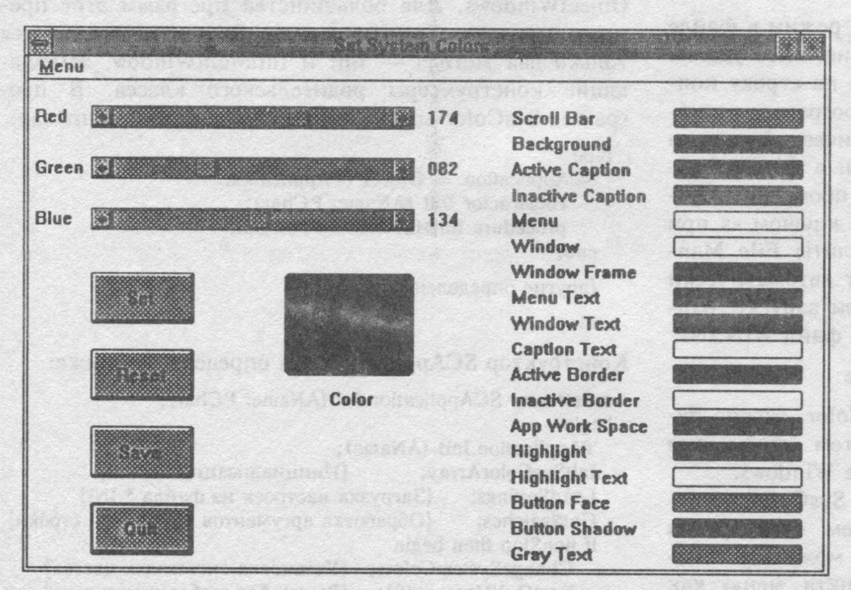


Рис. 2. Окно программы SysColor

При помощи кнопок управления полосами прокрутки вы можете изменять соотношения цветовых компонент, выбирая тем самым один из более чем 16 миллионов цветов. Большинство цветовых оттенков Windows изображает штриховкой, поскольку ни один

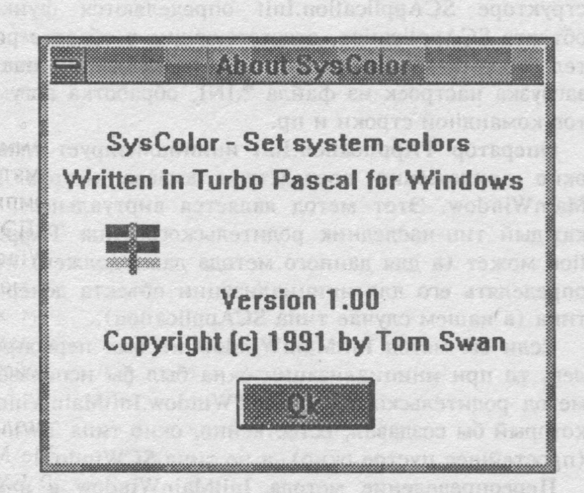


Рис. 3. Окно "About SysColor ..."

персональный компьютер не может отображать на дисплее такое количество цветов одновременно.

Поработав с программой SysColor, вы заметите, что она управляется только мышью. Без мыши вы можете вызвать программу и работать с меню, однако изменить системные цвета не можете. Конечно, программу можно усовершенствовать так, чтобы она поддерживала одновременно и работу с клавиатурой, но в данном варианте в целях простоты этого не предусмотрено.

Для того чтобы изменить системный цвет, укажите курсором мыши на одну из цветовых полос справа и нажмите на левую кнопку мыши. Не отпуская ее, переместите курсор на то поле, куда вы хотите скопировать этот цвет, и отпустите левую кнопку (если вы отпустите кнопку в поле окна вне цветовых полос, то копирование будет отменено). Цвета можно копировать из одной системной области в другую или из большого цветового поля в центре окна (цвет которого можно установить с помощью полос прокрутки).

Меню программы SysColor очень простое и содержит всего две команды: команду "About SysColor...", которая отображает окно описания программы, и команду "Exit", позволяющую завершить работу программы.

Диалоговое окно, вызываемое командой "About SysColor...", — стандартный объект Windows. Для его поддержки нужно минимальное количество операторов TPW (см. рис. 3 и процедуру SCWindow.CMAbout в листинге программы).

После изменения одного или нескольких цветов можно внести сделанные изменения в Windows, для чего нужно "нажать" мышью на кнопку **Set**. Для тех системных областей, где можно использовать только чистые цвета, Windows будет заменять комбинированные цвета на ближайшие чистые.

Выбранную вами настройку системных цветов можно сохранить в файле системных цветов Windows SYSCOLOR.INI. Для этого нужно "нажать" кнопку **Save**. Программа SysColor создаст этот файл в каталоге Windows. Если вы изменили системные цвета, но не сохранили их, при следующем запуске SysColor палитра системных цветов не будет соответствовать текущей. В этом случае можно сбросить неверную палитру кнопкой **Reset**.

Автоматическая загрузка палитры

Сохранив выбранную настройку в файле SYSCOLOR.INI, вы можете настроить программу SysColor таким образом, чтобы она при вызове загружала цвета

автоматически. Для этого при помощи текстового редактора добавьте в файл SYSCOLOR.INI строку

```
nonstop = true
```

После этого при следующем запуске SysColor программа автоматически загрузит цвета из файла SYSCOLOR.INI и "нажмет" кнопки **Set** и **Quit**. Запустить программу SysColor в автоматическом режиме можно также из командной строки при помощи ключа -n:

```
SYSCOLOR -n
```

Если вы установили автоматический режим в файле SYSCOLOR.INI, то отключить его можно, либо удалив строку **nonstop = true**, либо заменив ее на строку **nonstop = false**. Можно также запустить программу с ключом -s, который отключает автоматический режим даже в том случае, если он установлен в файле SYSCOLOR.INI. Удобнее всего задать для программы SysColor постоянную командную строку с ключом -s при помощи команды File | Properties утилиты File Manager. Для того чтобы утилита SysColor автоматически загружалась и устанавливала цвета при запуске Windows, следует добавить команду RUN в файл WIN.INI:

```
run = C:\WINDOWS\syscolor.exe
```

При этом, вызывая программу SysColor, можно будет менять текущие настройки, с тем чтобы они использовались при следующем запуске Windows.

Обратите внимание, что программа SysColor позволяет устанавливать цвета лучше, чем стандартная утилита Color. Так, с ее помощью вы можете установить цвет выделения выбранного пункта меню как черный на фоне морской волны:

```
Highlight = RGB (0, 255, 255)
   Highlight Text = RGB (0, 255, 255)
```

Как работает программа SysColor

Знакомство с исходным текстом программы SysColor лучше всего начинать с ее конца, где вы можете увидеть следующие строки:

```
var
      SCApp: SCApplication;

begin
   SCApp.Init (app_Name);
   SCApp.Run;
   SCApp.Done
end.
```

Если вы знакомы с TurboVision — объектно-ориентированным окружением для программирования на языке Turbo Pascal под MS-DOS — то легко узнаете эти операторы. Все они встречаются в программах на TurboVision точно в таком же виде (за исключением параметра функции SCApp.Init, который указывает имя программы). Однако не следует делать вывод о том, что TurboVision и библиотека ObjectWindows TPW совместимы друг с другом. Нет, они не совместимы, хотя идеология ObjectWindows во многом сходна с TurboVision и имена многих объектов этих библиотек совпадают друг с другом. Для того чтобы перенести программу TurboVision в ObjectWindows, придется проделать довольно значительную работу. И, тем не менее, если вы знакомы с TurboVision, вам значительно легче будет освоить ObjectWindows.

Как и в TurboVision, большинство программ, использующих окружение ObjectWindows, начинаются с описания объекта, принадлежащего классу TApplication — самому старшему классу в иерархии объектов ObjectWindows. Для большинства программ этот производный класс весьма прост. В нем определяются только два метода — Init и InitMainWindow, замещающие конструкторы родительского класса. В программе SysColor создание нового класса выглядит так:

```
type
  SCApplication = Object (TApplication)
     constructor Init (AName: PChar);
     procedure InitMainWindow; virtual;
  end;

  {другие определения}

end;
```

Конструктор SCApplication.Init определяется ниже:

```
constructor SCApplication.Init(AName: PChar);
begin
  TApplication.Init (AName);
  InitSysColorArray;          {Инициализация цветов}
  LoadSettings;     {Загрузка настроек из файла *.INI}
  GetSwitches;      {Обработка аргументов командной строки}
  if nonStop then begin
     ChangeSystemColors;  {Установка системных цветов}
     PostQuitMessage(0);   {Выход без отображения окна}
  end;
end;
```

В этом конструкторе первый оператор TApplication.Init (AName) вызывает стандартный метод Init объекта родительского типа (TApplication). В нем реализована стандартная последовательность инициализации окна приложения для Windows. Затем в конструкторе SCApplication.Init определяются функции объекта SCApplication, отсутствующие в объекте родительского типа TApplication: инициализация цветов, загрузка настроек из файла *.INI, обработка аргументов командной строки и пр.

Оператор TApplication.Init инициализирует главное окно приложения посредством вызова метода InitMainWindow. Этот метод является виртуальным, т.е. каждый тип-наследник родительского типа TApplication может (а для данного метода даже должен) переопределять его для инициализации объекта дочернего типа (в нашем случае типа SCApplication).

Если бы метод InitMainWindow не был переопределен, то при инициализации окна был бы использован метод родительского класса TWindow.InitMainWindow, который бы создавал, естественно, окно типа TWindow (простейшее пустое окно), а не типа SCWindow.

Переопределение метода InitMainWindow в нашем случае выглядит так:

```
procedure SCApplication.InitMainWindow;
begin
  MainWindow : =
          New (PSCWindow, Init (nil, 'Set System Colors'));
end;
```

Функция **New** резервирует свободное место в глобальном хипе для окна — объекта типа SCWindow, создаваемого конструктором этого типа SCWindow.Init. Поскольку это окно не принадлежит никакому окну данного приложения, первым параметром, передаваемым конструктору SCWindow.Init, является **nil**. Если бы создавалось дочернее окно, принадлежащее другому (родительскому) окну, то первым параметром вместо нуля следовало бы передавать адрес родительского объекта (обычно хранящийся в параметре **Self**). Вторым параметром передается текстовая строка, которая будет отображена в заголовке окна.

Если вы посмотрите на описание типа SCWindow, то обнаружите, что это сложный объект. Именно здесь определены почти все переменные, процедуры и функции программы SysColor. Для облегчения восприятия этого объекта, его описание в исходном тексте разделено на три части: данные, замещаемые методы родительского класса и новые методы. Пока, наверное, не следует углубляться в изучение этого объекта. Предположим, что окно создано и процедура SCApp.Init завершила свою работу. Что происходит дальше?

Возвращаясь к основной программе из трех строк, можно увидеть, что после инициализации приложения вызывается функция SCApp.Run. Те из вас, кто уже программировал для Windows, могут высказать предположение, что именно здесь находится цикл обработки сообщений (Message Loop). Однако, как это ни странно, программы на ObjectWindows не имеют цикла обработки сообщений. Более того, нигде в программе SysColor вы не найдете ничего похожего на обычные механизмы диспетчеризации сообщений Windows, так хорошо знакомые программистам на C! Как же тогда программа обрабатывает события? Для того чтобы это понять, необходимо разобраться с новой для системы Turbo Pascal особенностью — использованием Динамических Методов (Dynamic Methods).

Динамические методы и обработка событий

Операционная оболочка Microsoft Windows является средой, управляемой событиями. События (перемещения мыши, нажатия ее кнопок и клавиш на клавиатуре и пр.) обрабатываются оболочкой Windows и переводятся в системные сообщения, которые помещаются сначала в системную очередь сообщений, а потом направляются в очередь приложения, которому это сообщение предназначено (или, в отдельных случаях, всем приложениям). Так, например, когда вы нажимаете левую клавишу мыши, Windows посылает приложению сообщение, кодируемое константой wm_MouseMove.

Те из вас, кто программирует на C, узнали знакомое сообщение, которое на C выглядит как WM_MOUSEMOVE. Конечно, имена всех системных констант и флагов, используемых в TPW, совпадают с определенными в SDK. Однако в Pascal не принято вводить константы большими буквами (а имена сообщений являются в TPW константами), и поэтому имена сообщений не печатаются большими буквами, как в SDK. Поскольку Pascal, в отличие от C и C++, не различает большие и малые буквы, на совместимость с SDK это небольшое изменение не влияет. Зная, что константы в TPW начинаются всегда с малой буквы, вы никогда не спутаете константу с процедурой или функцией (а также с объектом или указателем, имена которых начинаются с букв "T" и "P" соответственно). Если вы будете пользоваться этими несложными общепринятыми соглашениями, вы повысите читабельность своих программ и сможете легче разобраться в чужих.

Что же происходит с сообщением, которое Windows посылает приложению? В обычном (не объектно-ориентированном) программировании программа сама извлекает сообщение из очереди приложения и передает соответствующей подпрограмме. Программируя на TPW, вы избавлены от необходимости писать код на таком низком уровне (хотя в принципе это возможно). ObjectWindows берет на себя всю обработку сообщений. Обычно этого бывает достаточно для большинства применений.

Для обработки сообщений в ObjectWindows имеются специальные методы. Для каждого сообщения существует свой метод. По приходе сообщения в очередь приложения WindowsObject вызывает для его обработки соответствующий метод. При таком способе обработки сообщений объекты приложения реагируют на сообщения независимо друг от друга (возможно, даже одновременно).

В определении типа SCWindow, наследника типа TWindow, метод-обработчик, например, сообщения wm_MouseMove может быть определен следующим образом:

```
SCWindow = Object (TWindow)
  ...
  procedure WMMouseMove (var Msg: TMessage);
       virtual wm_First + wm_MouseMove;
  ...
end;
```

При вызове метода **WMMouseMove** в параметр **Msg** помещается указатель на запись стандартного типа **TMessage**, содержащую всю информацию сообщения. Так, в случае сообщения **wm_MouseMove** через параметр **Msg** в определении метода доступны координаты **X** и **Y** нового положения мыши.

Имя метода WMMouseMove отвечает принятому соглашению об именах методов обработки сообщений. По этому имени ясно, что метод обрабатывает сообщение Windows (Windows Message, WM) wm_MouseMove. Однако вы можете назвать метод обработки этого сообщения и по-другому — **не имя метода определяет тип обрабатываемого сообщения.** Как же ObjectWindows определяет, какой метод вы-

звать при приходе сообщения wm_MouseMove в очередь приложения?

В ObjectWindows эта диспетчеризация осуществляется при помощи механизма так называемых *динамических методов*. Каждому методу обработки сообщения ставится в соответствие уникальный индекс в специальной таблице. В приложении, переопределяя методы обработки сообщений, программист также присваивает каждому методу свой индекс. Так, в примере выше в строке, начинающейся с оператора virtual, индекс метода WMMouseMove определяется как wm_First + wm_MouseMove. При этом константа wm_First определяет стандартный индекс начала таблицы динамических методов всех оконных сообщений Windows, а константа wm_MouseMove — смещение в этой таблице, по которому осуществляется передача управления по приходе сообщения wm_MouseMove. Аналогично константа cm_First используется для установки индекса метода на начало таблицы методов — обработчиков командных сообщений (сообщений, приходящих при выборе команд меню). При этом определение метода обработки командных сообщений отличается от определения метода обработки оконных сообщений только индексом таблицы динамических методов. Вот, например, как выглядит в программе SysColor определение метода обработки команды "About SysColor ..." меню программы:

```
SCWindow = Object (TWindow)
...
procedure CMAbout (var Msg: TMessage);
   virtual cm_First + cm_About;
...
end;
```

Значение константы cm_About здесь то же, что и в определении меню программы SysColor в файле ресурсов.

Эта константа прибавляется к константе cm_First и используется в качестве индекса для доступа к методу через таблицу динамических методов. Этим однозначно определяется, что при выборе пункта "About SysColor ..." меню SysColor будет вызвана функция CMAbout. Отметим, что, поскольку выражения, определяющие индексы, являются константами, они вычисляются на этапе компиляции и не замедляют работу программы.

Комбинирование обычного и объектного программирования

Программа на TPW не обязана быть полностью "объектно-ориентированной". Как показывают первые шесть процедур программы SysColor, вы можете применять смешанное — объектно-ориентированное и традиционное программирование. Из исходных текстов видно также, что вы можете в любое время вызывать различные функции Windows для выполнения тех или иных операций. Таким образом, нет никакой необходимости оформлять каждую операцию Windows в виде объекта.

Например, для того чтобы загрузить в массив программы набор системных цветов, процедура InitSysColorArray непосредственно вызывает функцию GetSysColor библиотеки WinProcs, а процедура ChangeSystemColors аналогичным образом вызывает функцию SetSysColors. Справочное руководство по TPW подробно описывает эти и другие функции Windows, которыми можно пользоваться в программах на TPW; общее число их достигает 600 (!).

Просматривая исходный текст SysColor, программисты на Turbo Pascal для MS-DOS сразу обратят внимание на несколько необычных для Pascal конструкций, служащих для обеспечения совместимости программ на Pascal с системой, написанной на С, каковой является Microsoft Windows. Первое, что бросается в глаза, — это новые функции обработки строк, которые в С должны оканчиваться нулями. В Turbo Pascal для Windows включена специальная библиотека Strings, которая содержит процедуры и функции для копирования, сравнения и выполнения других стандартных операций над такими строками, а также преобразования строк из формата Pascal в формат С и обратно. Одна из таких процедур — StrPCopy вызывается в функции Int2Str программы SysColor. Она переводит строку Pascal "S" в оканчивающуюся нулем строку формата С, указатель на которую помещает в переменную "С" типа PChar.

Тип PChar — это также нововведение для Pascal. Переменная типа PChar является указателем на массив символов, оканчивающийся нулем. Первый элемент такого массива имеет индекс 0 (а не 1, как это принято в Pascal). Таким образом, переменная типа PChar является указателем на символьную строку, принятую в языке С и используемую в функциях Microsoft Windows. Если вызываемая функция Windows требует, чтобы аргумент был дальним указателем на строку символов (LPSTR в Microsoft SDK, что соответствует типу char far * языка С), то передаваемая строка должна быть приведена к типу PChar, например:

```
Attr.Menu := LoadMenu (HInstance, PChar (id_Menu));
```

Наиболее проницательные из вас непременно заметят, что в приведенном выше примере параметр id_Menu не является строкой — это целочисленная константа, определяющая индекс меню. Как же в этом случае работает преобразование типа PChar (id_Menu), и вообще, что происходит при передаче такого параметра функции LoadMenu системы Windows? Результат этого преобразования можно представить таким образом:

Переменная	Dec	Hex
id_Menu	100	$0064
PChar (id_Menu)	–	$0000:0064

Как видим, функции LoadMenu на самом деле верный указатель не передается (строка приложения

пользователя не может храниться в нулевом сегменте), а идентификатор меню записывается в смещении передаваемого фиктивного адреса.

Этот трюк связан с необычной возможностью, которой обладают некоторые функции Windows. Если функции Windows требуется передать идентификатор ресурса ("resource ID"), всегда имеющий тип PChar (LPSTR в SDK), то вы можете передать ей как имя идентификатора ресурса (текстовую строку), так и сам идентификатор (целое число). Если в сегментной части адреса находится не ноль, то считается, что передается адрес текстовой строки; если в сегментной части ноль (фиктивный адрес), то функция извлекает из смещения непосредственно сам идентификатор. Второй вариант намного короче, поскольку при этом нет необходимости извлекать идентификатор из таблицы ресурсов.

Возможно, программисту на Pascal эти ухищрения покажутся чрезмерными, но, как известно, в чужой монастырь со своим уставом не ходят: программируя для системы Windows, написанной на C, приходится придерживаться соглашений и стиля программирования языка C. Переписать же библиотеку Windows на язык Pascal невозможно: дело в том, что библиотека Windows — это часть операционной оболочки (вызовы функций Windows отрабатываются самой Windows). Так что программистам на Pascal, по-видимому, придется смириться со странными с точки зрения Pascal конструкциями типа PChar (id_Menu).

Интерфейс с графическими устройствами

Ни одна статья о программировании в Windows не может быть полной без упоминания об интерфейсе с графическими устройствами (Graphics Device Interface, GDI) — богатой библиотеке структур данных, функций и процедур, обеспечивающих графический вывод на т.н. *виртуальное графическое устройство*. Для любителей графики эта библиотека делает программирование в Windows особенно привлекательным. Вместо того чтобы создавать несколько вариантов одной и той же программы для совместимости с различными типами устройств, вы можете написать одну-единственную программу, которая при помощи GDI будет гарантированно осуществлять графический вывод на все графические устройства — от принтеров и плоттеров до мониторов высокого разрешения. Вообще, если вы используете GDI, то ваша программа сможет работать на любом устройстве, которое поддерживает GDI.

В программе SysColor используется несколько функций GDI, но это только крохотная часть колоссального набора графических процедур. Большинство функций очень просты. Посмотрите, например, на процедуру **DrawRubberBand**, рисующую "резиновую линию", которую можно видеть при копировании цветов. В этой и других графических функциях вы увидите параметр **DC** или **PaintDC**. Эти параметры определяют структуру Windows, которая называется также *контекстом отображения* (display context). Это довольно сложное понятие GDI, включающее в себя описание физических свойств конкретного устройства и, кроме того, использующееся для разделения рабочих областей различных приложений Windows.

В многооконной операционной оболочке, какой является Windows, экран монитора — разделяемый ресурс, т.е. он может использоваться одновременно многими приложениями. В каждый конкретный момент времени приложение Windows "не знает", какую область экрана оно занимает. Поэтому очевидно, что графический вывод в Windows не может осуществляться непосредственно на экран (например, путем записи в видеопамять) — эту функцию выполняет операционная оболочка, работающая с контекстами отображения различных приложений.

Органы управления и дочерние окна

У объекта класса SCWindow, наследника класса TWindow, в свою очередь, есть дочерние объекты (окна). Одним из таких дочерних объектов является, например, полоса прокрутки для изменения веса красного цвета. Кнопка **Set** также является дочерним объектом главного окна программы SysColor. Такие дочерние окна называются *органами управления*.

Как это известно всем пользователям Windows, органы управления обычно располагаются в специальных диалоговых окнах. Но, как показывает пример программы SysColor, органы управления можно также разместить непосредственно на поле главного (или любого другого) окна приложения Windows.

В SysColor обращение к трем цветовым полосам прокрутки осуществляется посредством указателей:

```
SBarRed, SBarGrn, SBarBlu : PScrollBar;
```

При помощи этих указателей можно обращаться к объектам типа **TScrollBar** — одного из нескольких стандартных типов объектов — органов управления, определенных в ObjectWindows. К другим стандартным органам управления относятся, например, **TButton** (кнопка), **TCheckBox** (кнопка "Вкл/Выкл"), **TRadioButton** (кнопка-переключатель) и другие. То, что эти и другие органы управления оформлены в виде объектов, значительно упрощает работу с ними.

Для того чтобы к органам управления можно было обращаться, их адреса помещаются в локальных переменных (**SBarRed**). Сами объекты создаются всего лишь одним оператором:

```
SBarRed := New (PScrollBar,
        Init(@Self, id_SBarRed, 50, 20, 250, 0, true));
```

Конструктор инициализирует объект и вызывает процедуры Windows, которые создают соответствующие структуры данных этого органа управления. Аргумент **@Self** передает адрес текущего объекта SCWindow как адрес родительского объекта (на "языке" Windows дочерний объект принадлежит родительскому). Другие аргументы определяют сообщение, которое генерирует орган управления (**id_SBarRed**),

координаты органа управления в окне приложения и горизонтальную ориентацию полосы прокрутки **true** (если бы передавался аргумент **false**, то полоса прокрутки была бы ориентирована вертикально).

Для того чтобы обрабатывать события полосы прокрутки, в классе **SCWindow** объявляется специальный метод обработки сообщения **id_SBarRed**. Объявление этого метода весьма похоже на объявление методов обработки других сообщений; отличается только индекс динамического метода:

```
procedure SBarRedEvent (var Msg: TMessage);
    virtual id_First + id_SBarRed;
```

Подобно индексам **wm_First** и **cm_First** константа **id_First** определяет начальное значение индекса таблицы динамических методов органов управления. Сумма **id_First + id_SBarRed** определяет уникальный индекс, который ObjectWindow использует для вызова метода по получении сообщения **id_SBarRed**. В результате, когда происходит какое-либо событие с полосой прокрутки, метод **SBarRedEvent** работает так, как если бы он был физически соединен со "своим" органом управления. Обрабатывая событие, метод может извлекать информацию об органе управления из сообщения **Msg**. Например, для того чтобы извлечь значение, на которое в данный момент времени установлен регулятор полосы прокрутки, в методе **SBarRedEvent** используется следующий оператор:

```
RedColor := SBarRed^.GetPosition;
```

Другие органы управления и поддерживающие их методы создаются и обрабатываются аналогично. Как можно убедиться из листинга программы SysColor, большая часть кода, реализующего функции программы SysColor, приходится на долю именно методов поддержки дочерних органов управления — типичная ситуация для программы на Windows.

Подведем итоги

Программа SysColor дает начальное представление о том, как можно писать программы в среде ObjectWindows. Я работаю с TPW уже год и получил ни с чем не сравнимое удовольствие наблюдать за тем, как компилятор рос от нескольких простых тестовых файлов до полномасштабного средства разработки программ для Windows, серьезно претендующего на лидерство в этой области. Программа SysColor была одной из моих ранних попыток написать полное приложение для Windows. Поскольку среда была для меня сравнительно новая, создание программы заняло больше времени, чем можно было предполагать. И, тем не менее, легкость, с которой я осваивал технологию ObjectWindows, произвела на меня довольно сильное впечатление.

И все же я не утверждаю, что программировать на TPW с использованием окружения ObjectWindows просто. Создание приложений для Windows — сложная задача, вне зависимости от того, с какой системой программирования вы работаете. Однако по прошествии нескольких месяцев использования TPW и ObjectWindows я не смог найти причины, которая заставила бы меня вернуться к SDK и языку C. TPW предлагает свежий объектно-ориентированный подход к программированию для Windows, и для меня нет дороги назад.

Т.Сван

По материалам:
Borland Language Express, 1991, Vol., №2
Статью перевел и переработал *С.Гладков*

```
{ SYSCOLOR.PAS — Set System Colors (c) 1991 by Tom Swan.}
{$R SYSCOLOR.RES}

program SysColor;

uses WinTypes, WinProcs, WObjects, Strings;

const
  app_name    = 'SysColor';          { Имя приложения }
  ini_FName   = 'SYSCOLOR.INI';      { Имя файла настроек, .INI }
  id_Menu     = 100;                 { Идентификатор ресурса }
  id_Icon     = 200;                 { Идентификатор пиктограммы }
  cm_About    = 101;                 { Идентификатор команды "About ..." }
  cm_Quit     = 102;                 { Идентификатор команды "Exit" }
  id_SBarRed  = 100;                 { Идентификаторы органов управления }
  id_SBarGrn  = 101;
  id_SBarBlu  = 102;
  id_STxtRed  = 103;
  id_STxtGrn  = 104;
  id_STxtBlu  = 105;
  id_SetBtn   = 106;
  id_ResetBtn = 107;
  id_SaveBtn  = 108;
  id_QuitBtn  = 109;
  RedMask     = $000000FF;           { Маска извлечения значения цвета }
  GrnMask     = $0000FF00;
  BluMask     = $00FF0000;
  nonStop: Boolean = false;          { Ключи: -s = отмена; -n = установка }
  SysColorName: Array[0..color_EndColors] of PChar = (
    'Scroll Bar',
    'Background',
    'Active Caption',
    'Inactive Caption',
    'Menu',
    'Window',
    'Window Frame',
    'Menu Text',
    'Window Text',
    'Caption Text',
    'Active Border',
    'Inactive Border',
    'App Work Space',
    'Highlight',
    'Highlight Text',
    'Button Face',
    'Button Shadow',
    'Gray Text',
    'Button Text'
  );
type
  SCApplication = Object(TApplication)
    constructor Init (AName: PChar);
    procedure InitMainWindow; virtual;
  end;
  PSCWindow = ^SCWindow;
  SCWindow = Object (TWindow)
    { Поля данных CSWindow }
    Dc                              : Hdc;
    ButtonDown, Changed             : Boolean;
    LineX1, LineY1, LineX2, LineY2  : Integer;
```

```
    ArrowCursor, CrossHairCursor        : HCursor;
    RedColor, GrnColor, BluColor        : Byte;
    SBarRed, SBarGrn, SBarBlu           : PScrollBar;
    STxtRed, STxtGrn, STxtBlu           : PStatic;
    SampleRect                          : TRect;
    SampleColor                         : TColorRef;
    DraggingOrigin                      : Integer;
    { Наследуемые CSWindow методы }
    constructor Init (AParent: PWindowsObject; ATitle: PChar);
    function    CanClose: Boolean; virtual;
    procedure   GetWindowClass (var AWndClass: TWndClass); virtual;
    procedure   SetupWindow; virtual;
    procedure   WMLButtonDown (var Msg: TMessage);
                virtual wm_First+wm_LButtonDown;
    procedure   Paint (PaintDC: HDC; var PaintInfo: TPaintStruct);
                virtual;
    { Новые методы CSWindow }
    function    InsideColorRect (X, Y: Integer; var Index: Integer): Boolean;
    procedure   ResetSystemColors;
    procedure   SynchronizeScrollBars;
    procedure   DrawRubberBand;
    procedure   CMAbout (var Msg: TMessage);
                virtual cm_First+cm_About;
    procedure   CMQuit  (var Msg: TMessage);
                virtual cm_First+cm_Quit;
    procedure   WMLButtonUp (var Msg: TMessage);
                virtual wm_First+wm_LButtonUp;
    procedure   WMMouseMove (var Msg: TMessage);
                virtual wm_First+wm_MouseMove;
    procedure   SBarRedEvent (var Msg: TMessage);
                virtual id_First+id_SBarRed;
    procedure   SBarGrnEvent (var Msg: TMessage);
                virtual id_First+id_SBarGrn;
    procedure   SBarBluEvent (var Msg: TMessage);
                virtual id_First+id_SBarBlu;
    procedure   SetBtnEvent (var Msg: TMessage);
                virtual id_First+id_SetBtn;
    procedure   ResetBtnEvent (var Msg: TMessage);
                virtual id_First+id_ResetBtn;
    procedure   SaveBtnEvent (var Msg: TMessage);
                virtual id_First+id_SaveBtn;
    procedure   QuitBtnEvent (var Msg: TMessage);
                virtual id_First+id_QuitBtn;
  end;
  SysColorRec = record
    OriginalColor  : LongInt;      { Цвет при вызове программы }
    CurrentColor   : LongInt;      { Новый установленный цвет }
    SCRect         : TRect;        { Координаты цветового прямоугольника }
  end;
var
  SysColorArray: Array [0..color_EndColors] of SysColorRec;

  {------Общие процедуры------}

  { Преобразование целого N в строковый массив C.
    Если Max > 0, заполнить нулями. }
  procedure Int2Str (N, Max: Integer; C: PChar);
  var S: String [6];
  begin
    Str (N, S);
    while Length (S) < Max do S := `0` + S;
    StrPCopy (C, S);
  end;

  { Заполнить глобальный массив Syscolor текущими цветами }
  procedure InitSysColorArray;
  var
    i: Integer;
  begin
    for i := 0 to color_EndColors do
      with SysColorArray [i] do begin
        OriginalColor := GetSysColor (i);
        CurrentColor  := OriginalColor;
        with SCRect do begin
          Left   := 500;
          Top    := 20 + (i * 20);
          Right  := Left + 100;
          Bottom := Top + 15;
        end;
      end;
  end;

  { Изменение системных цветов }
  procedure ChangeSystemColors;
  var
    i: Integer;
    InxArray: Array [0..color_EndColors] of Integer;
    ClrArray: Array [0..color_EndColors] of TColorRef;
  begin
    for i := 0 to color_EndColors do begin
      InxArray[i] := i;
      ClrArray[i] := SysColorArray[i].CurrentColor
    end;
    SetSysColors (color_EndColors+1, InxArray[0], ClrArray);
  end;

  { Сохранение цветов в файле SYSCOLOR.INI в каталоге Windows }
  function SaveSettings: Boolean;
  var
    i: Integer;
    S: String [12];
    NewValue: Array [0..12] of Char;
  begin
    SaveSettings := true;
    for i := 0 to color_EndColors do
      with SysColorArray [i] do begin
        Str(CurrentColor, S);
        StrPCopy (NewValue, S);
        if not WritePrivateProfileString (app_Name,
               SysColorName [i], NewValue, ini_FName) then begin
          SaveSettings := false;
          Exit
        end;
      end;
  end;

  { Загрузка цветов из файла SYSCOLOR.INI (если есть) }
  procedure LoadSettings;
  var
    i, Err: Integer;
    S:      String [12];
    DefaultValue, NewValue: Array [0..12] of Char;
  begin
    for i := 0 to color_EndColors do
      with SysColorArray[i] do begin
        Str(CurrentColor, S);
        StrPCopy (DefaultValue, S);
        GetPrivateProfileString (app_Name, SysColorName [i],
               DefaultValue, NewValue, sizeof (NewValue), ini_FName);
        S := StrPas (NewValue);
        Val (S, CurrentColor, Err);
        if Err <> 0 then CurrentColor := OriginalColor;
      end;
    GetPrivateProfileString (app_Name, `nonstop`,
          `false`, NewValue, sizeof (NewValue), ini_FName);
    if StrComp (`false`, NewValue) <> 0 then
      nonStop := true;
  end;

  { Извлечение ключей командной строки }
  procedure GetSwitches;
  var
    i: Integer;
    S: String [128];
    C: Char;
  begin
    for i := 1 to ParamCount do begin
      S := ParamStr (i);
      C := upcase (S[1]);
      if (Length (S) > 1) and ((C = `-`) or (C = `/`)) then
      case upcase (S[2]) of
        `N`: nonStop := true;
        `S`: nonstop := false;
      end;
    end;
  end;

  {--Методы SCApplication--}

  { Создание объекта SCApplication }
  constructor SCApplication.Init(AName: PChar);
  begin
    TApplication.Init (AName);
    InitSysColorArray;                  { Инициализация цветов }
    LoadSettings;                       { Загрузка начальных настроек }
    GetSwitches;                        { Обработка ключей командной строки }
    if nonStop then begin
      ChangeSystemColors;               { Изменение цветов по *.INI }
      PostQuitMessage(0);               { Немедленный выход }
```

```
      end;
  end;

{ Инициализация окна приложения }
procedure SCApplication.InitMainWindow;
begin
  MainWindow := New(PSCWindow, Init (nil, 'Set System Colors'));
end;

{--Методы SCWindow--}

{ Создание объекта SCWindow и дочерних окон }
constructor SCWindow.Init(AParent: PWindowsObject; ATitle: PChar);
var
  AStat: PStatic;
  ABtn:  PButton;
begin
  TWindow.Init (AParent, ATitle);
  Attr.Menu := LoadMenu (HInstance, PChar (id_Menu));
  with Attr do begin
    X := 10; Y := 10; H := 460; W := 615;
  end;
  ButtonDown := false;
  Changed := false;
  ArrowCursor := LoadCursor (0, idc_Arrow);
  CrossHairCursor := LoadCursor (0, idc_Cross);
  RedColor := 0;
  GrnColor := 0;
  BluColor := 0;
  SampleColor := 0;
  with SampleRect do begin
    Left := 200; Top := 150; Right := 300; Bottom := 230;
  end;
  SBarRed := New (PScrollBar, Init(@Self, id_SBarRed, 50, 20, 250, 0, true));
  SBarGrn := New (PScrollBar, Init(@Self, id_SBarGrn, 50, 60, 250, 0, true));
  SBarBlu := New (PScrollBar, Init(@Self, id_SBarBlu, 50, 100, 250, 0, true));
  AStat := New(PStatic, Init(@Self, 0, 'Red', 5, 20, 40, 20, 3));
  AStat := New(PStatic, Init(@Self, 0, 'Green', 5, 60, 40, 20, 5));
  AStat := New(PStatic, Init(@Self, 0, 'Blue', 5, 100, 40, 20, 4));
  AStat := New(PStatic, Init(@Self, 0, 'Color', 235, 240, 40, 20, 5));
  STxtRed := New(PStatic, Init(@Self, id_STxtRed, '000', 310, 20, 40, 20, 3));
  STxtGrn := New(PStatic, Init(@Self, id_STxtGrn, '000', 310, 60, 40, 20, 3));
  STxtBlu := New(PStatic, Init(@Self, id_STxtBlu, '000', 310, 100, 40, 20, 3));
  ABtn := New(PButton, Init(@Self, id_SetBtn, 'Set', 50, 150, 80, 40, false));
  ABtn := New(PButton, Init(@Self, id_ResetBtn, 'Reset', 50, 210, 80, 40, false));
  ABtn := New(PButton, Init(@Self, id_SaveBtn, 'Save', 50, 270, 80, 40, false));
  ABtn := New(PButton, Init(@Self, id_QuitBtn, 'Quit', 50, 330, 80, 40, true));
end;

{ Если окно может закрыться, возвратить true }
function SCWindow.CanClose: Boolean;
var Answer: Integer;
begin
  CanClose := true;
  if Changed then begin
    Answer := MessageBox (HWindow, 'Save Colors before quitting?',
       'Please answer', mb_YesNoCancel or mb_IconQuestion);
    if Answer = idYes then CanClose := SaveSettings
    else if Answer = idCancel then CanClose := false;
  end;
end;

{ Переустановка системных цветов на ранее сохраненные }
procedure SCWindow.ResetSystemColors;
var
  i: Integer;
begin
  for i := 0 to color_EndColors do with SysColorArray [i] do
    CurrentColor := OriginalColor;
  Changed := false;
end;

{ Изменение класса окна для регистрации пиктограммы }
procedure SCWindow.GetWindowClass (var AWndClass: TWndClass);
begin
  TWindow.GetWindowClass (AWndClass);
  AWndClass.hIcon := LoadIcon (HInstance, PChar (id_Icon))
end;

{ Начальная настройка объекта SCWindow }
procedure SCWindow.SetupWindow;
begin
  TWindow.SetupWindow;
  SBarRed^.SetRange(0, 255);
  SBarGrn^.SetRange(0, 255);
  SBarBlu^.SetRange(0, 255)
end;

{ Установка полос прокрутки по выбранному цвету }
procedure SCWindow.SynchronizeScrollBars;
var
  DummyMsg: Tmessage;
begin
  SBarRed^.SetPosition (SampleColor and RedMask);
  SBarGrn^.SetPosition ((SampleColor and GrnMask) shr 8);
  SBarBlu^.SetPosition ((SampleColor and BluMask) shr 16);
  SBarRedEvent (DummyMsg);
  SBarGrnEvent (DummyMsg);
  SBarBluEvent (DummyMsg);
end;

{ Отображение диалогового окна "About" }
procedure SCWindow.CMAbout (var Msg: TMessage);
var
  Dialog: TDialog;
begin
  Dialog.Init (@Self, 'About');
  Dialog.Execute;
  Dialog.Done;
end;

{ Выполнение команды 'Exit' }
procedure SCWindow.CMQuit (var Msg: TMessage);
begin
  PostQuitMessage(0);
end;

{ "Резиновая линия", отображаемая при копировании цвета }
procedure SCWindow.DrawRubberband;
begin
  MoveTo (Dc, LineX1, LineY1);
  LineTo (Dc, LineX2, LineY2);
end;

{ Если точка (X, Y) находится в цветовом прямоугольнике, вернуть true }
function SCWindow.InsideColorRect (X, Y: Integer; var Index: Integer): Boolean;
var
  CursorLocation: TPoint;
  i: Integer;
begin
  CursorLocation.X := X;
  CursorLocation.Y := Y;
  if PtInRect (SampleRect, CursorLocation) then begin
    Index := -1;                                  {Внутри прямоугольника}
    Exit
  end else
    for i := 0 to Color_EndColors do
      if PtInRect(SysColorArray[i].SCRect, CursorLocation) then begin
        Index := i;
        Exit
      end;
  InsideColorRect := false;
end;

{ Обработка нажатия левой клавиши мыши }
procedure SCWindow.WMLButtonDown (var Msg: TMessage);
begin
 if not ButtonDown then with Msg do
   if InsideColorRect (LParamLo, LParamHi, DraggingOrigin)
   then begin
     Dc    := GetDC (HWindow);
     LineX1 := LParamLo;
     LineY1 := LParamHi;
     LineX2 := LineX1;
     LineY2 := LineY1;
     SetROP2 (Dc, r2_Not);
     DrawRubberBand;
     ButtonDown := true;
     SetCursor (CrossHairCursor);
     SetCapture (HWindow);
     if DraggingOrigin >= 0 then begin
       SampleColor := SysColorArray[DraggingOrigin].CurrentColor;
       SynchronizeScrollBars
     end
   end
end;

{ Обработка отпускания левой клавиши мыши }
procedure SCWindow.WMLButtonUp(var Msg: TMessage);
var
  Index: Integer;
  NewColor: TColorRef;
begin
  if ButtonDown then with Msg do begin
```

```
    if InsideColorRect (LParamLo, LParamHi, Index) then
      if (Index < > DraggingOrigin) and (Index > = 0) then begin
        Changed : = true;
        if DraggingOrigin > = 0 then
          NewColor : = SysColorArray[DraggingOrigin].CurrentColor
        else
          NewColor : = SampleColor;
        SysColorArray[Index].CurrentColor : = NewColor;
        InvalidateRect (HWindow, nil, false)
      end;
    DrawRubberBand; {Erase last line}
    SetROP2 (Dc, r2_Black);
    ButtonDown : = false;
    SetCursor (ArrowCursor);
    ReleaseDC (HWindow, Dc);
    ReleaseCapture
  end
end;

{ Отработка перемещения мыши }
procedure SCWindow.WMMouseMove(var Msg: TMessage);
begin
  if ButtonDown then begin
    DrawRubberband;               {Удаление старой линии}
    with Msg do begin
      LineX2 : = LParamLo;
      LineY2 : = LParamHi;
      DrawRubberband              {Отрисовка новой линии}
    end
  end
end;

{ Отработка события в красной полосе прокрутки}
procedure SCWindow.SBarRedEvent (var Msg: TMessage);
var
C: Array [0..3] of Char;
begin
  RedColor : = SBarRed^.GetPosition;
  Int2Str(RedColor, 3, C);
  STxtRed^.SetText(C);
  SampleColor : = RGB (RedColor, GrnColor, BluColor);
  InvalidateRect (HWindow, @SampleRect, false)
end;

{ Отработка события в зеленой полосе прокрутки }
procedure SCWindow.SBarGrnEvent (var Msg: TMessage);
var
C: Array [0..3] of Char;
begin
  GrnColor : = SBarGrn^.GetPosition;
  Int2Str(GrnColor, 3, C);
  STxtGrn^.SetText(C);
  SampleColor : = RGB (RedColor, GrnColor, BluColor);
  InvalidateRect (HWindow, @SampleRect, false)
end;

{ Отработка события в голубой полосе прокрутки }
procedure SCWindow.SBarBluEvent (var Msg: TMessage);
var
C: Array [0..3] of Char;
begin
  BluColor : = SBarBlu^.GetPosition;
  Int2Str(BluColor, 3, C);
  STxtBlu^.SetText(C);
  SampleColor : = RGB (RedColor, GrnColor, BluColor);
  InvalidateRect (HWindow, @SampleRect, false)
end;

procedure SCWindow.SetBtnEvent (var Msg: TMessage);
begin
  ChangeSystemColors
end;

procedure SCWindow.ResetBtnEvent (var Msg: TMessage);
begin
  ResetSystemColors;
  ChangeSystemColors
end;

procedure SCWindow.SaveBtnEvent (var Msg: TMessage);
begin
  if SaveSettings then Changed : = false;
end;

procedure SCWindow.QuitBtnEvent (var Msg: TMessage);
begin
  PostQuitMessage (0);
end;
procedure SCWindow.Paint (PaintDC: HDC; var PaintInfo: TPaintStruct);
var
  OldBrush, TheBrush  : HBrush;
  i                   : Integer;
procedure ShowSysColor (i: Integer);
var
  SysColorBrush       : HBrush;
  OldBrush            : HBrush;
  SCName              : PChar;
begin
  with SysColorArray[i], SCRect do begin
    SysColorBrush : = CreateSolidBrush (CurrentColor);
    OldBrush : = SelectObject (PaintDC, SysColorBrush);
    RoundRect (PaintDC, Left, Top, Right, Bottom, 5, 5);
    SelectObject (PaintDC, OldBrush);
    DeleteObject (SysColorBrush);
    SCName : = SysColorName [i];
    TextOut (PaintDC, Left - 125, Top, SCName, StrLen (SCName))
  end
end;

begin
  TheBrush : = CreateSolidBrush (SampleColor);
  OldBrush : = SelectObject (PaintDC, TheBrush);
  with SampleRect do
    RoundRect (PaintDC, Left, Top, Right, Bottom, 10, 10);
  SelectObject (PaintDC, OldBrush);
  DeleteObject (TheBrush);
  for i : = 0 to color_EndColors do
    ShowSysColor(i)
end;

var
SCApp: SCApplication;
begin
  SCApp.Init (app_Name);
  SCApp.Run;
  SCApp.Done
end.
```

```
{ SYSCOLOR.RC, (c) 1991 by Tom Swan.}

# include < E:\TPW\OWL\windows.h >

200 ICON syscolor.ico
100 MENU LOADONCALL MOVEABLE PURE DISCARDABLE
BEGIN
  POPUP "&Menu"
  BEGIN
    MenuItem "&About syscolor...", 101
    MenuItem "E&xit",              102
  END
END

ABOUT DIALOG DISCARDABLE LOADONCALL PURE MOVEABLE 29,48,128,99
STYLE WS_POPUP | WS_CAPTION | WS_SYSMENU | 0x80L
CAPTION "About SysColor"
BEGIN
  CONTROL 200                                              101,
    "STATIC", WS_CHILD | WS_VISIBLE | 0x3L,                14, 36, 17, 17
  CONTROL "SysColor - Set system colors"                   106,
    "STATIC", WS_CHILD | WS_VISIBLE,                       15, 11, 99, 10
  CONTROL "&Ok"                                            1,
    "BUTTON",WS_CHILD|WS_VISIBLE|WS_TABSTOP,               49, 77, 35, 13
  CONTROL "Written in Turbo Pascal for Windows"            107,
    "STATIC", WS_CHILD | WS_VISIBLE,                       4, 21, 120, 10
  CONTROL "Copyright (c) 1991 by Tom Swan"                 108,
    "STATIC", WS_CHILD | WS_VISIBLE,                       14, 63, 105, 10
  CONTROL "Version 1.00"                                   109,
    "STATIC", WS_CHILD | WS_VISIBLE,                       45, 52, 44, 9
END
```

Новый компилятор фирмы Borland — Borland C++ 2.0 имеет все средства, необходимые для создания Windows-программ, и позволяет обходиться без Microsoft SDK.

Компилятор Borland C++ 2.0. Создание Windows-программ без SDK

До недавнего времени на рынке инструментальных программных средств было всего несколько компиляторов языка С, поддерживающих разработку Windows-программ. Эти компиляторы — Microsoft C и Zortech C++ — при создании Windows-программ требовали использования Microsoft Windows Software Development Kit (SDK).

С появлением Borland C++ 2.0 (BCPP) ситуация изменилась. BCPP имеет все средства, необходимые для разработки Windows-программ; использования SDK при этом не требуется.

С одной стороны, BCPP — это новый, более развитый представитель семейства компиляторов Turbo C/Turbo C++. С другой стороны, он является зачинателем новой группы компиляторов C/C++, поддерживающих разработку Windows-программ. По сообщениям фирмы, в дальнейшем будут развиваться оба семейства продуктов — компиляторы, работающие в среде DOS (Turbo C++), и компиляторы, работающие в среде Windows (Borland C++). Последние будут ориентированы преимущественно на разработку Windows-программ.

Интегрированная среда разработчика (ИСР) BCPP, уже знакомая пользователям Turbo C++, соответствует рекомендациям стандарта SAA/CUA (System Application Architecture/Common User Access) фирмы IBM и поддерживает многооконное редактирование файлов, работу с мышью, текстовый редактор с широкими возможностями, а также имеет встроенный отладчик. Помимо этого, в комплект BCPP входит отладчик Turbo Debugger (TD) с интерфейсом, похожим на ИСР компилятора. Turbo Debugger позволяет отображать исходный текст программы, содержимое регистров процессора и памяти и устанавливать точки прерываний. С помощью TD можно отлаживать программы, созданные при помощи компиляторов как фирмы Borland, так и фирмы Microsoft. Используя входящий в комплект профилировщик Turbo Profiler (TF), можно анализировать производительность программы и определять те участки, которые требуют оптимизации. Как и компилятор Turbo C++, BCPP содержит два компилятора. Один из них — компилятор С, поддерживающий стандарт ANSI C, другой — компилятор С++, поддерживающий спецификацию AT&T версии 2.0. Помимо компилятора с интегрированной средой поставляется пакетный компилятор.

Кроме упомянутых, в комплект BCPP входят еще два компилятора (BCX и BCCX), работающие в защищенном режиме процессора и позволяющие использовать при компиляции программ всю доступную расширенную память.

Компилятор BCPP поддерживает встроенный ассемблер, не требующий, как в предыдущих версиях, загрузки Turbo Assembler или Microsoft Macro Assembler для компиляции программ. Также в состав BCPP входят библиотеки классов С++, включая потоки и контейнеры (связанные списки, стеки, очереди и т.п.).

Но самой замечательной возможностью нового компилятора фирмы Borland является поддержка создания Windows-программ.

Компилятор

Компилятор BCPP стремится полностью заменить компилятор Microsoft C (MSC), поэтому он более совместим с MSC, чем предыдущие компиляторы фирмы Borland. Давайте сравним эти два компилятора.

Примерно половина функций библиотеки Borland C++ имеет те же названия, что и у MSC. Это связано в первую очередь с тем, что оба компилятора поддерживают стандарт ANSI C. Ряд функций, включенных в библиотеку Borland C++, выполняют те же действия, имеют те же формальные параметры, что и функции библиотеки Microsoft C, но наименования параметров и функций BCPP и MSC различны. В заголовочных файлах BCPP имеется ряд макросов для поддержки вызовов MSC-функций с преобразованием их в вызовы функций BCPP. Для облегчения компилятору вызова этих функций можно использовать макрос __MSC.

Ряд заголовочных файлов BCPP и MSC имеют различные наименования, но в большинстве своем они совместимы. Различия в наименованиях файлов заголовков приведены в таблице 1.

Таблица 1. Различия в наименованиях файлов заголовков

BCPP 2.0	MSC 6.0
ALLOC.H	MALLOC.H
DIR.H	DIRECT.H
MEM.H	MEMORY.H

Для компиляции Windows-программ в BCPP используется макрос _Windows, позволяющий определять, какие Windows-функции необходимы при компиляции.

Как уже было сказано, BCPP поддерживает встроенный ассемблер. Для включения в текст программы ассемблерных вставок в MSC используется ключевое слово _asm:

```
_asm {
  mov ax,1
  xor bx,bx
}
```

В BCPP можно применять как ключевое слово _asm, так и ключевое слово asm:

```
_asm mov ax,1
```

или:

```
asm mov ax,1
```

Помимо встроенного ассемблера для обращения к регистрам в BCPP, как и в Turbo C 2.0, и Turbo C++, можно использовать псевдорегистровые переменные:

```
_AX = 2;
```

Создание Windows-программ

Как отмечалось выше, в состав BCPP входят четыре компилятора. Два из них работают с интегрированной средой, два — с командной строкой. Два компилятора (один с интегрированной средой и один пакетный) работают в реальном режиме процессора, остальные два — в защищенном режиме процессоров 80286 и выше. Все компиляторы позволяют создавать Windows-программы, работающие в защищенном режиме (в стандартном и расширенном режимах Windows). Компиляторы не создают кода для работы DOS-программ в защищенном режиме.

Компиляция в защищенном режиме позволяет использовать до 15 Мбайт памяти под буферы и создаваемые объектные файлы, тем самым уменьшая число обращений к диску и существенно повышая скорость компиляции. Компиляция в защищенном режиме возможна на машинах с процессорами 80286 и выше и 640 Кбайтами оперативной памяти. Для работы компилятора BCX требуется 576 Кбайт расширенной или дополнительной памяти, а для компилятора BCCX — 450 Кбайт.

Как известно, для успешного создания Windows-программ компилятор должен поддерживать:

- соглашение о вызовах типа Pascal (параметры передаются в стек слева направо, и подпрограмма сама очищает стек по завершении работы);
- ключевые слова near и far соответственно для 16-битных внутрисегментных ссылок и 32-битных межсегментных ссылок (статические данные Windows-программ доступны через указатели типа near, а указатели на функции Windows должны быть типа far);
- генерацию специального пролога и эпилога.

Компилятор BCPP удовлетворяет всем перечисленным требованиям и помимо Windows-функций также позволяет создавать динамически загружаемые библиотеки (DLL).

Так как файлы заголовков, используемые при создании Windows-программ, имеют довольно большие размеры (один файл WINDOWS.H больше 120 Кбайт), очень удобно использование реализованной в BCPP возможности создания *предкомпилированных файлов заголовков*. В процессе компиляции файла заголовков на диск записывается таблица символов. При повторной компиляции программы файл заголовков не компилируется, а загружается уже готовая таблица. Необходимость предкомпиляции файлов заголовков указывается опцией командной строки компиляторов BCC и BCCX:

```
BCC -H или BCCX -H
```

или командой Compiler|Code Generation|Precompiled Headers команды меню Options ИСР. В первом случае создается файл TCDEF.SYM, а во втором — файл PROJECT.SYM, где PROJECT — имя текущего проекта.

Таблица 2. Время компиляции, с

Компиляция	MSC 6.0A	BCC	BCCX	BCC*	BBCX*
1	7.63	5.79	5.29	4.84	5.82
2**	5.92	4.57	4.56	2.42	2.73

* Использовались предкомпилированные файлы заголовков.
** Компиляция с использованием расширенной памяти.

Опции MSC 6.0A:
CL -Gsw -AS winhello.c /link libw.lib slibcew.lib
Опции BC++:
Без предкомпиляции:
BCC -WS winhello.c
BCCX -WS winhello.c
С учетом предкомпиляции:
BCC -H -WS winhello.c
BCCX -H -WS winhello.c

Как видно из приведенной таблицы, при повторной компиляции с использованием предкомпилированных файлов заголовков время компиляции сокращается почти в два раза.

Из той же таблицы видно, что компиляция, использующая расширенную память, выполняется существенно быстрее.

Средство для создания и редактирования ресурсов — Whitewater Resource Toolkit

Ресурсы Windows-программ — это блоки диалога, графические изображения, курсоры, иконки, меню и строки текста. Существует возможность динамического создания ресурсов во время выполнения программы, но чаще всего ресурсы создаются с помощью специальных редакторов и подключаются к исполняемому файлу. В отличие от динамически создаваемых ресурсов, статические (подключенные к .EXE-файлу) ресурсы загружаются в память только при необходимости и в дальнейшем могут быть выгружены из памяти. Если одновременно запущено несколько копий одной и той же программы, то все они будут использовать только одну копию ресурса в памяти. Использование статических ресурсов существенно облегчает локализацию программных продуктов (перевод интерфейсной части прикладной программы на другой язык).

Включенный в состав компилятора BCPP редактор ресурсов Whitewater Resource Toolkit (WRT) позволяет создавать и редактировать Windows-ресурсы. В состав WRT включены редакторы для всех стандартных ресурсов — каждый из них представлен иконкой-кнопкой в основном окне WRT. WRT также имеет два окна для просмотра ресурсов в файлах (.RES, .DLL или .EXE), копирования ресурсов, их переименования и включения ресурсов прямо в .EXE-файлы. WRT создает файлы, совместимые с компилятором ресурсов RC.EXE (входящим в состав как Microsoft SDK, так и BCPP), так что этот компилятор можно использовать для записи ресурсов в .EXE-файлы. В большинстве случаев использования компилятора ресурсов RC.EXE не требуется.

Редактор ресурсов WRT разработан с таким расчетом, чтобы заменить утилиты SDKPaint и Dialog, входящие в состав Microsoft SDK. Функционально эти средства одинаковы, но WRT использует более наглядный, интуитивный интерфейс и имеет ряд расширений, например позволяет тестировать создаваемые меню. Недостатком WRT является то, что с его помощью невозможно редактировать шрифты.

Turbo Debugger for Windows

В состав BCPP входят несколько отладчиков; среди них — Turbo Debugger и Turbo Debugger for Windows (TDW). Отладчик Turbo Debugger для DOS-программ хорошо известен программистам. TDW является новинкой и позволяет отлаживать Windows-программы. TDW может работать на двухмониторной системе или в качестве полноэкранной DOS-задачи в среде Windows. TDW работает в текстовом режиме и позволяет по нажатии Alt-F5 просматривать текущий экран Windows. Отладка Windows-программ требует несколько иного подхода, чем отладка обычных DOS-программ. Вместо пошагового выполнения (которое в случае Windows обычно приводит в цикл работы с сообщениями) используются точки останова по сообщениям (messages) Windows. С помощью TDW можно указать как отдельное сообщение, так и целый класс сообщений (сообщения от мыши, клавиатуры, системные сообщения и т.п.). По сообщению можно остановить выполнение программы или записать это сообщение в специальном окне.

TDW позволяет просматривать как локальный, так и глобальный хип, загруженные задачи и DLL-библиотеки, отлаживать динамические библиотеки.

Возможна отладка программ, созданных с помощью компилятора MSC. Для этого необходимо с помощью утилиты TDCONVERT преобразовать символьные таблицы из формата MSC в формат Borland.

TDW и Turbo Debugger предоставляют уникальную возможность выполнения программы на несколько шагов назад.

К недостаткам TDW относится некорректная работа отладчика с некоторыми видеоадаптерами (в частности, по завершении работы портится экран на видеоадаптерах SuperVGA типа Video7).

Turbo Assembler и Turbo Profiler

Новая версия ассемблера Turbo Assembler 2.5 поддерживает более гибкий интерфейс с языками высокого уровня. Несколько директив (MODEL, PROC, EXTRN, PUBLIC, COMM, GLOBAL, PUBLICDLL) поддерживают спецификатор языка высокого уровня (Pascal, BASIC, FORTRAN, C, Prolog). Также можно указать спецификатор языка в инструкции CALL, что приведет к генерации кода в соответствии с соглашениями о вызовах для данного языка высокого уровня (передача параметров и очистка стека).

Директива ассемблера .MODEL позволяет указать спецификатор WINDOWS. Это вызовет автоматическую генерацию Windows-пролога и эпилога для данной подпрограммы. Директива PUBLICDLL указывает, что процедура является точкой входа в DLL.

В состав BCPP также включена версия ассемблера, работающая в защищенном режиме процессора. Эта версия позволяет компилировать программы большого объема.

В отличие от MSC, где компилятор сам выполняет необходимую оптимизацию (часто существенно увеличивая время компиляции), в состав BCPP включен профилировщик Turbo Profiler, позволяющий вручную оценить производительность программы. Turbo Profiler дает возможность посмотреть время выполнения каждой подпрограммы и выявить "узкие места", которые в наибольшей степени влияют на общую производительность.

Документация

В комплект поставки BCPP включены девять книг документации и справочная система. В документации подробно описаны различные аспекты, начиная от установки компилятора на жесткий диск, использования ИСР и кончая справочником по стандартной библиотеке и руководствами по ассемблеру, отладчику и профилировщику. Файлы с исходными текстами программ, представленных в документации в качестве примеров, также включены в комплект поставки. Справочная система, доступная из ИСР или вызываемая при помощи утилиты THELP, содержит материалы по компилятору, стандартной библиотеке, интерфейсу Windows Application Program Interface, а также описание структур данных, сообщений и констант Windows. На диске имеется файл, вкратце описывающий порядок создания Windows-программ с помощью C++.

Несмотря на заверения фирмы о полном отказе от Microsoft SDK, для успешного создания Windows-программ с помощью Borland C++ требуется как минимум пособие, посвященное программированию в среде Windows (подобное тому, что входит в состав Microsoft SDK), или книга Ч.Петзольда.

Заключение

Компилятор Borland C++ имеет все средства, необходимые для создания Windows-программ с использованием C, C++ или ассемблера. Ряд фирм заявили о выпуске библиотек классов для создания Windows-программ.

Совсем недавно фирма Borland выпустила библиотеку классов ObjectWindows, позволяющую упростить создание объектно-ориентированных Windows-программ. В состав библиотеки входят классы, ориентированные на создание элементов интерфейса — диалоговых блоков, меню и т.п. Библиотека также осуществляет поддержку потокового ввода/вывода. Поставляется исходный текст библиотеки.

А.Федоров

Компьютеризация биржевой деятельности идет полным ходом

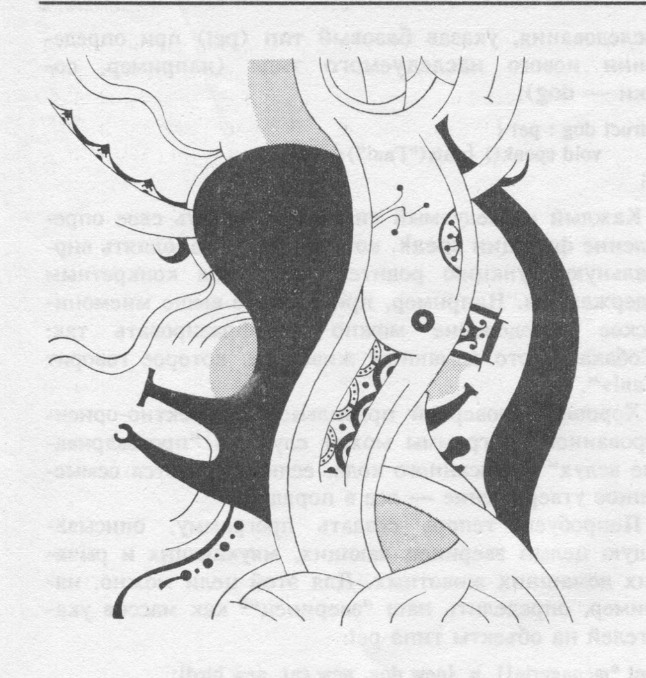
Биржи мира планируют отказаться от своих залов и полностью перевести торги в он-лайновый режим. Быстрее всего этот процесс идет в области торговли валютой. Reuters начали испытания в реальном режиме своей системы Dealing 2000 в Лондоне и Нью-Йорке. Работать она начнет в начале следующего года. Dealing 2000 заключает сделку автоматически, если цены спроса и предложения совпадают. В ответ на это Dow Jones продаст свою биржевую службу Trading Service, чтобы стать владельцем более крупной доли в дилерской системе Minex, которая перейдет в он-лайновый режим в 1993 году. Основными акционерами Minex являются KDD, Токийская валютная биржа и японские банки. В результате предпринимаемых действий Dow Jones войдет в тройку самых крупных держателей акций Minex.

Кроме того, нью-йоркская биржа Mercantile Exchange подписала соглашение о создании Access, биржевой он-лайновой системы, работающей тогда, когда обычные биржи закрыты, для заключения фьючерных сделок по нефти и металлам. Система начнет работу в конце 1992 года.

The Teleputing Hotline, November 12, 1991

DOW JONES представила службу составления подборок под названием Facts Delivered. Она представляет собой поток новостей из информационных каналов компании, отобранных в соответствии с указанной тематикой, которые могут направляться на факсимильные аппараты или в электронные почтовые ящики. Подписчики платят 15 долларов в месяц за подбор материала, и еще 1.50 доллара за каждые 1000 символов текста ($1.80 если вы не в США).

Teleputing Hotline, November 14, 1991



Для того чтобы понять, нужно ли вам читать эту статью, проведем небольшой тест: когда вы в последний раз использовали оператор **virtual**? *Если ответ "никогда", то располагайтесь поудобнее и приступайте к чтению.*

Виртуальные функции? Это очень просто!

Конечно, ваши программы могут превосходно работать и без оператора virtual, и вам не обязательно понимать его смысл при работе с C++ или Turbo Pascal. Однако без этого оператора нельзя в полном объеме использовать возможности объектно-ориентированного программирования (ООП), а особенно создания новых типов данных.

В этой статье я хочу подвести вас к пониманию одной из ключевых особенностей ООП — использованию виртуальных функций — на примере одной маленькой программы. Хотя программа написана на C++, вы можете получить такой же результат и на Turbo Pascal.

Основное различие между традиционными языками и ООП — возможность создавать принципиально новые типы данных.

Каждый тип данных в ООП имеет свои внутренние характеристики (способ хранения информации, длину в байтах), а также определяет операции, которые можно выполнять над объектами этого типа. Например, в языке C число с плавающей точкой имеет порядок (показатель степени), мантиссу и бит знака; с ним можно производить различные арифметические действия (складывать, вычитать и т.п.). В ООП вы можете создать свои собственные типы данных, которые будут обрабатываться так же, как и встроенные типы: компилятор будет выполнять контроль соответствия типов, вы сможете создавать массивы новых типов и т.д. Таким образом, добавляя свои собственные типы данных, вы фактически расширяете средство, на котором работаете.

Какие же преимущества в таком расширении языка? С помощью ООП вы можете привести среду программирования в соответствие с условиями решаемой задачи вместо того, чтобы приспосабливать постановку задачи к имеющимся средствам программирования. Другими словами, предметная область непосредственно отображается в область решения. Поэтому создавать и эксплуатировать большие и очень сложные программы в ООП становится намного легче и дешевле: сокращаются сроки разработки программ и уменьшаются используемые ресурсы.

В C++ новые типы данных являются расширением структур (**struct**), которые могут теперь содержать не только данные, но и функции. Переменная, заданная такой структурой, называется **объектом**. При этом

каждая переменная-объект содержит свои, уникальные данные, а код функции, определенной в структуре, компилятор создает только один раз. Область действия такой функции ограничена элементами структуры, в которой эта функция определена. Компилятор C++ создает новый тип данных (с именем структуры) при помощи механизма, аналогичного **typedef**. (Вы можете также использовать оператор **class**, действующий подобно **struct**, но обладающий некоторыми отличиями. Однако сейчас мы не будем на этом останавливаться.)

В качестве примера рассмотрим текст программы PETS.CPP, которая моделирует некий дом, полный домашних животных[1].

Прежде всего мы должны описать основной (родительский) тип данных, назовем его **pet**, который будет объединять в себе самое общее свойство домашних животных. Пусть общей характеристикой всех животных будет то, что они издают различные звуки (предположим, что рыбки издают очень тихие звуки)[2]:

```
struct pet {
      virtual void speak() {}
};
```

В структуре **pet** определена одна-единственная функция **speak**. Эта функция не выполняет в данном случае никаких действий. Ее присутствие определяет одно из общих свойств домашних животных — по сути, функция просто объявляет: "Домашние животные имеют свойство говорить (speak)". Для любого объекта типа pet вы можете вызвать функцию speak. Что же касается данных, общих для всех домашних животных, то в качестве таковых можно определить, например, возраст и название животного (**int age** и **char *name**).

Обратите внимание на то, что функция speak описывается в месте определения. Это так называемая **inline function** — встроенная (в код) функция. Такое определение похоже на макроопределение препроцессора — компилятор помещает в исполняемый код не заголовок вызова функции, а код реализации самой функции везде, где бы она ни вызывалась. Причем при вызове компилятор осуществляет полную проверку типа функции. Заметьте также, что в нашем случае тип **pet** не имеет **конструктора** — специальной функции, выполняющей инициализацию объекта. На самом деле, если вы не помещаете конструктор явно, компилятор сам создает его и вызывает при размещении объекта в памяти (важность этого станет более очевидной позднее).

Как же теперь создать объекты **различных** животных (собак, кошек, птиц и т.д.), которые бы могли издавать характерные для них звуки? Это можно сделать с помощью реализованного в языке механизма наследования, указав **базовый** тип (pet) при определении нового **наследуемого** типа (например, собаки — **dog**):

```
struct dog : pet {
      void speak() {puts("Гав!");}
};
```

Каждый наследуемый тип может давать свое определение функции **speak**, которое будет наполнять виртуальную функцию родительского типа конкретным содержанием. Например, приведенное выше мнемоническое определение можно перефразировать так: "Собака — это домашнее животное, которое говорит «Гав!»".

Хорошей проверкой правильности объектно-ориентированной программы может служить "проговаривание вслух" написанного кода: если получается осмысленное утверждение — все в порядке!

Попробуем теперь создать программу, описывающую целый зверинец лающих, мяукающих и рычащих домашних животных. Для этой цели можно, например, определить наш "зверинец"[3] как массив указателей на объекты типа pet:

```
pet *menagerie[] = {new dog, new cat, new bird};
```

Это определение очень простое, однако оно требует многих разъяснений. При создании такого массива на стандартном С вы должны сначала зарезервировать место для его элементов, а потом поочередно инициализировать эти элементы. В языке C++ ничего этого не требуется — компилятор инициализирует переменные автоматически, для чего сам создает нужный код (число элементов массива в данном случае также определяется автоматически). Такая инициализация называется **совокупной инициализацией (aggregate initialization)** — инициализация объекта в памяти (его динамическое размещение) с последующей инициализацией данных. Поскольку основную работу мы возложили на компилятор, размер исходного текста программы уменьшается, она становится более читабельной и мы в итоге делаем меньше ошибок. В стандартном С тоже предусмотрены механизмы подобной инициализации (правда, в более примитивной форме), но при этом в списке инициализации могут быть только константы. В нашей же программе мы, вызывая оператор **new** для динамического размещения объектов в памяти, фактически выполняем код, создающий элементы списка инициализации. Оператор **new**, подобно функции **malloc()**, создает новый объект, резервируя под него место в оперативной памяти (а именно, в хипе — незанятом объеме памяти), и возвращает указатель на вновь созданный объект (размер резервируемой области определяется автоматически по типу объекта). Возвращаемый указатель автоматически включается в массив механизмом совокупной инициализации.

Способ вычисления **sz** (числа элементов массива) не нов — то же можно сделать и в стандартном С:

---

[1] Pet — ручное домашнее животное (англ.).

[2] То, какие именно общие свойства следует определить, решает программист в соответствии с постановкой задачи. В самом деле, мы могли бы выбрать в качестве общего свойства проживание на планете Земля или то, что все животные имеют определенные геометрические размеры.

[3] Menagerie — зверинец (англ.).

```
const int sz = sizeof(menagerie) / sizeof(menagerie[0]);
```

Этим только подтверждается факт, что массив menagerie весьма похож на обычный массив указателей: вы можете добавить в него новый элемент просто расширением списка инициализации. Обратите все же внимание на то, что вычисление sz осуществляется во время компиляции, а полученное значение включается прямо в генерируемый код, так что приведенная строка не вносит в программу дополнительных накладных расходов. В стандартном (ANSI) С оператор const определяет переменную, значение которой инициализируется при ее создании и не изменяется в процессе работы программы (переменная доступна только для чтения). В С++ оператор const работает аналогично директиве препроцессора #define с той лишь разницей, что при обращении к переменной осуществляется проверка соответствия типов.

Теперь, наконец, в функции main() вы можете заставить "говорить" всех "животных" "зверинца":

```
void main(void) {
  for (int i = 0; i < sz; i++)
     menagerie[i]->speak();
}
```

Здесь функция speak вызывается для каждого "животного" поочередно. В С++ на функции объекта ссылаются таким же образом, как на элементы данных в структурах языка С. Стрелкой (->) ссылаются на элемент через указатель

```
menagerie[i]->speak();
```

точкой — через сам объект

```
Fido.speak();
```

Замечу, в объектно-ориентированном программировании не принято говорить "вызвать функцию speak — свойство объекта Fido типа dog"; говорят иначе — "послать сообщение speak объекту Fido типа dog" (мы как бы просим объект Fido произвести действие; при этом способность осуществлять такие действия является свойством всех объектов типа dog).

Поскольку все "животные" могут "говорить" (это свойство определено в основном типе pet), то в этом отношении с ними всеми следует обращаться одинаково. Это чрезвычайно удобно: вы можете заставить любой объект данного типа произвести указанное действие, не обращая внимания на конкретную реализацию действия. Программа становится значительно проще. Кроме того, вы легко можете добавлять в "зверинец" новые типы "животных" без какой-либо модификации ранее написанного кода.

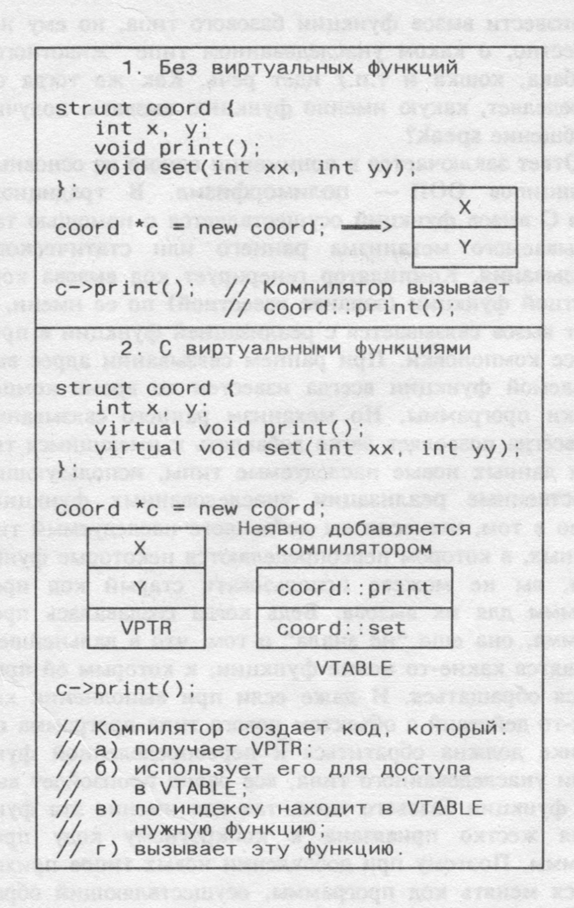
Какой speak вызвать?

Вот тут-то при более детальном размышлении и возникает проблема. Компилятору известно, что "зверинец" (menagerie) — это массив указателей на "животных" (объекты типа pet). Когда вы посылаете сообщение speak объекту типа pet, компилятор имеет информацию, достаточную только для того, чтобы произвести вызов функции базового типа, но ему неизвестно, о каком унаследованном типе "животного" (собака, кошка и т.п.) идет речь. Как же тогда он определяет, какую именно функцию вызвать, получив сообщение speak?

Ответ заключается в понимании одного из основных принципов ООП — полиморфизма. В традиционном С вызов функций осуществляется с помощью так называемого механизма раннего или статического связывания. Компилятор генерирует код вызова конкретной функции (заранее известной) по ее имени, и этот вызов связывается с реализацией функции в процессе компоновки. При раннем связывании адрес вызываемой функции всегда известен во время компоновки программы. Но механизм раннего связывания не всегда позволяет легко добавлять к имеющимся типам данных новые наследуемые типы, использующие собственные реализации унаследованных функций. Дело в том, что если вы добавляете наследуемый тип данных, в котором переопределяются некоторые функции, вы не можете использовать старый код программы для их вызова. Ведь когда создавалась программа, она еще "не знала" о том, что в дальнейшем появятся какие-то новые функции, к которым ей придется обращаться. И даже если при выполнении каких-то действий с объектом нового типа программа по логике должна обратиться к переопределенной функции унаследованного типа, все равно произойдет вызов функции базового типа, так как именно эта функция жестко привязана к конкретному коду программы. Поэтому при добавлении новых типов приходится менять код программы, осуществляющий обращения к функциям, которые в этих новых типах переопределены, и затем проводить повторную компиляцию всей программы.

Принцип полиморфизма (множественности образов), реализованный в языке С++, предлагает гибкий механизм для решения означенной проблемы — позднее или динамическое связывание с использованием специальных функций, называемых виртуальными (virtual). Механизм позднего связывания позволяет передавать сообщение о вызове виртуальной функции тому объекту, над которым выполняется действие. При этом виртуальная функция не связывается с объектом так, как это происходит с обычными функциями в процессе компиляции. Вы можете создать новый тип, унаследованный от базового, и написать для него другую подходящую функцию с тем же именем, теми же формальными параметрами (сигнатурой функции) и типом возвращаемого значения, что и у виртуальной функции объекта базового типа. Затем вы можете скомпилировать свой код и скомпоновать его с уже готовыми объектными (.OBJ) файлами кода функций, обращающихся к виртуальным функциям объектов базового типа[4]. Вызов необходимой функции (для объек-

[4] Заметьте, эти объектные файлы не требуют изменения и перекомпиляции. Таким образом, у вас появляется возможность передавать или продавать свои собственные програм-



тов базового или унаследованного типа) будет осуществляться уже в процессе выполнения программы и будет полностью зависеть от типа объекта, участвующего в вызове. Хочу еще раз подчеркнуть, что при позднем связывании адрес функции, соответствующей данному объекту, определяется **во время выполнения программы**.

Хотя при знакомстве с объектно-ориентированным программированием в первую очередь на себя обращает внимание **инкапсуляция**, самое важное свойство ООП — возможность создавать новые типы данных. А работа с новыми типами данных невозможна без позднего связывания, реализуемого в языках C++ и Turbo Pascal посредством виртуальных функций.

Во время компиляции в C++ осуществляется проверка того, что функция определена и аргументы и возвращаемое значение имеют верный тип. Однако в момент компиляции неизвестно, какой именно код будет выполняться при вызове виртуальной функции. Поэтому на место реального вызова функции компилятор помещает в программу несколько специальных операторов. В процессе работы программы эти операторы извлекают указатель, используемый для вычисления адреса нужной функции, из самого объекта. По этому указателю, который в языке C++ называется **VPTR (Virtual PoinTeR)**, а в Turbo Pascal — **VMTP (Virtual Method Table Pointer)**, из специальной таблицы выбирается физический адрес вызываемой процедуры. Эта таблица, называемая в C++ **VTABLE** (**VMT** в Turbo Pascal), содержит адреса всех виртуальных функций. Каждый объект имеет свой указатель **VPTR**. При вызове виртуальной функции сам объект определяет, какой код должен выполняться. Это станет более понятным, если вы изучите диаграмму.

Обработка виртуальных функций компилятором

Чтобы вычислить физический адрес функции, механизм позднего связывания выполняет дополнительные, по сравнению с обычным, процедуры и поэтому работает немного медленнее. Некоторые языки ООП осуществляют только позднее связывание для всех процедур. Но при программировании на C++ и Turbo Pascal у вас есть выбор: механизм позднего связывания включается явным указанием оператора **virtual**; в противном случае используется стандартный вызов. Причем здесь действует еще одно правило. Если в базовом типе вы объявили функцию виртуальной и в наследуемом типе вы не переопределяете сигнатуру функции (формальные параметры) и тип возвращаемого значения, то в этом наследуемом типе такая функция считается виртуальной по умолчанию. Кстати, если вы решили переопределить виртуальную функцию в наследуемом типе, то вы не можете изменить один только тип возвращаемого значения. Нельзя определить две виртуальные функции с одинаковыми именами и сигнатурами, но с разными типами возвращаемых значений. Если же две функции с одинаковыми именами имеют разные аргументы, то эти функции тоже считаются разными, и механизм позднего связывания игнорируется.

Каким же образом и когда инициализируется механизм позднего связывания — создается и заполняется таблица VTABLE, и в объектах устанавливается нужный вход в нее? Эту работу выполняет **конструктор объекта**. В нашем примере, где конструктор не определен, компилятор сам создает конструктор. В Turbo Pascal вы должны явно создать и вызвать конструктор для инициализации указателя VMT, в противном случае при выполнении программы будет выдано сообщение об ошибке. В C++ конструктор автоматически вызывается при выполнении оператора **new** (C++ гарантирует, что конструктор будет вызван).

Как известно, стандартный (ANSI) C не имеет виртуальных функций. Вы могли бы в какой-то мере смоделировать их: определить **speak** в типе **pet** как указатель на функцию, создать для каждого дочернего типа свою функцию **speak** и в каждом объекте установить

мные продукты, например, библиотеки классов, в виде объектного кода и header-файлов с объявлениями типов. Этим вы сохраняете свои программные секреты, но не ограничиваете возможности своих пользователей по наращиванию и дальнейшему развитию продукта.

указатель **speak** на нужную функцию. Но это же масса работы! Как вы уже могли видеть, в С++ это делается намного проще и удобнее.

А теперь немного разомнемся. Включите компьютер, введите текст приведенной ниже программы и, используя встроенный отладчик среды или Turbo Debugger, посмотрите, как она будет выполняться в пошаговом режиме. Если вы захотите увидеть сгенерированный код, то компилируйте программу с ключом -S в командной строке компилятора С++ фирмы Borland. Чтобы определить затраты на организацию виртуальных функций, создайте цикл, осуществляющий большое количество вызовов, и используйте Turbo Profiler. Потом удалите оператор **virtual** и повторите тест.

В качестве заключительного эксперимента убедитесь, насколько просто можно расширить программу. Добавьте в “зверинец“ “животное“ нового типа, скажем, лошадь (horse). Наиболее интересный эффект получится в случае, если создать для этого отдельный модуль, куда включить и код инициализации массива. Вы увидите, как программа управляет объектом, о существовании которого она ранее не подозревала. Я уверен, что выполнив все это самостоятельно, вы поймете, как работают виртуальные функции, и по достоинству оцените их.

Б.Эккел

```
// Текст программы PETS.CPP
// Перекличка домашних животных

#include <stdio.h>

struct pet {                              // базовый тип
  virtual void speak() {}                 // определение функции
};

struct dog : pet {                        // наследуемые типы
  void speak() {puts(“Гав!“);}
};
struct cat : pet {
  void speak() {puts(“Мяу!“);}
};
struct bird : pet {
  void speak() {puts(“Чик-Чирик!);}
};
struct goldfish : pet {                   // goldfish — золотая рыбка
  void speak() {puts(“!“);}
};

// Совокупная инициализация “зверинца“
pet *menagerie[] = { new dog, new cat, new bird };
const int sz = sizeof(menagerie) / sizeof(menagerie[0]);

void main(void) {
  for (int i = 0; i < sz; i + + )
    menagerie[i] -> speak();
}
```

Куда мы направляемся

С тех пор, как мы начали издавать наш бюллетень в 1988 году, редакция Teleputing Hotline постоянно задавала себе вопрос — “куда приведет нас вся эта технология?“

Наш ответственный за распространение Hiro Nakamura ответил на этот вопрос самым непосредственным образом. Лет 5 назад он был сфотографирован в горах работающим на своем лэптоп-компьютере. Там не было воды, не было электричества, привязывающих его к городу. Но там была телефонная линия. Сегодня она бы ему не понадобилась.

Проводные телефонные сети имеют большую пропускную способность. Через них можно крутить телефильмы, передавать факс, звук, и видеосигнал одновременно или производить через них миллионы операций в минуту. Это все просто прекрасно, если вы работаете в офисе. Но две трети американских рабочих там не работают. В других странах соотношение еще хуже.

Как быть с ними? 90-е годы ответят на это простой концепцией “компьютеризации на выезде“ (Field Computing).

Миниатюрные компьютеры любых форм, размеров, с интерфейсами, удобными конкретному пользователю, будут связаны с компьютерными сетями без проводов и с минимальными затратами. Когда файл попадает в проводную сеть, он может быть чем угодно и делать что угодно. Не обязательно он будет письмом. Он может стать счетом, денежным переводом, заказом товаров или услуг. Не обязан он быть и “просто данными“. Это может быть все что угодно — изображение, оцифрованный звук или фильм в виде “видеопочты“.

Люди, пестующие эту технологию, заработают в грядущем десятилетии большие деньги, производя больше, чем конкуренты, и с меньшими затратами. Они будут предоставлять лучшие услуги. “Компьютеризация на выезде“ в 90-х годах станет тем же, чем персональные машины стали для 80-х.

Мы собираемся отслеживать эту технологию. С этого момента The Teleputing Hotline станет вашим источником информации о Field Computing. Следите за нами, учитесь у нас, следуйте примерам и зарабатывайте.

The Teleputing Hotline, November 7, 1991

Частоты в восточной Германии свободны для мобильных сетей

Советские войска выводятся из восточной Германии, и German Bundespost может использовать старые военные частоты для систем переносных телефонов PCN. Планируется выдать третью лицензию на мобильную телефонию для использования этих новых частот. Две фирмы, которые являются операторами в настоящее время, Mannesman Mobilfunk and Bundespost Telecom, могут не быть допущены до соискания, так как они недостаточно быстро развивались в восточном направлении. Новая служба может начать работу к концу 1992 года, сообщили немецкие власти.

The Teleputing Hotline, November 12, 1991

КОМПЬЮТЕРЫ САММИТ СИСТЕМС

EGA 640×350

VGA 640×480

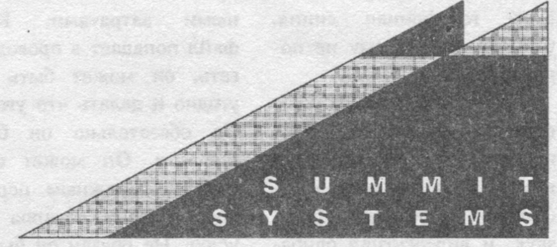
SuperVGA 1024×768

Сберечь глаза, сидя перед компьютером по 8 часов в день, - проблема. Саммит Системс - это *идеальное разрешение* ваших проблем!



- Возможность преобразования САММИТ-286 в САММИТ-386/33
- SuperVGA монитор 1024x768, 256 цветов
- Быстрый винчестер (17 мс; 105 / 52 Мб)
- ОЗУ до 8 Мб (286) и до 32 Мб (386)
- Местное обслуживание
- Гарантия 2 года

(095) 265-5813, 261-4407, 299-1162
(0172) 973-119, факс: 973-519



Поставьте Будущее Себе на Стол



шлет пламенный ПРИВЕТ всем своим нынешним поклонникам!
говорит ДОБРО ПОЖАЛОВАТЬ своим будущим пользователям!
уже начала свое победное шествие по необъятным просторам одной шестой части суши.

Вы хотите добиться успеха, применив в своем бизнесе новые информационные технологии? И Вы думаете, что Вам удастся осуществить это без ВИКТОРИИ? А Вы знаете о тех могущественных возможностях ВИКТОРИИ, обладателем которых можете стать ВЫ? Неужели Вы в состоянии от всего этого отказаться? Узнав, что у Вас есть ВИКТОРИЯ, Ваши конкуренты сдадутся без боя!

ВИКТОРИЯ — это волшебная палочка в Ваших руках!

Вы программист? Вы пользователь? Вы новичок?
Новое компьютерное поколение выбирает ВИКТОРИЮ!
Не упустите счастливый случай!

Вам нравится Norton Commander? PCTools? XTree?
Да ведь Вы еще не работали с ВИКТОРИЕЙ!

ВИКТОРИЯ – это Ваша СИЛА
ВИКТОРИЯ – это Ваше МОГУЩЕСТВО
ВИКТОРИЯ – это Ваше ПРЕИМУЩЕСТВО
Ваш ИНТЕЛЛЕКТ и Ваша ВИКТОРИЯ сделают Вас НЕПОБЕДИМЫМ!

И Вы все еще сомневаетесь, какую оболочку Вам выбрать?

ВИКТОРИЯ доступна всем! Мы поддерживаем предельно низкие цены!

ВИКТОРИЯ — это атомная бомба в Вашем компьютере

ВИКТОРИЯ – это новая SOFT-БОМБА!

ВИКТОРИЯ – это ваша рабочая лошадка!

У Вас есть ВИКТОРИЯ! Ваши конкуренты в панике!

Работать с ВИКТОРИЕЙ – хороший тон для бизнесмена!

Вы приобрели ВИКТОРИЮ! Победа у Вас в кармане!

Phones: (095) 420-83-80
(095) 491-01-53



Появление новой операционной системы — всегда очень важное и знаменательное событие, которого с нетерпением ожидают не столько пользователи, сколько те, кто так или иначе причастен к компьютерной индустрии.

Разрешите представиться: MS-DOS 5.0!

Долгожданная DOS 5.0, о выпуске которой сегодня пишет вся мировая компьютерная пресса, впервые поступила в продажу в июле 1991 года. Тогда же пришли первые сообщения о том, что фирмы IBM и Compaq приобрели право выпуска собственных модификаций DOS 5.0, а сотни фирм помельче уже получили права на выпуск этой операционной системы по лицензии для поставки в составе своих компьютерных систем. Многие фирмы-производители прикладного программного обеспечения откликнулись выпуском своих новых продуктов, обеспечивающих работу в среде новой операционной системы. Одними из первых были фирмы Symantec с сервисным пакетом Norton Utilities 6.0 и Central Point Software с конкурирующим пакетом PC Tools 7.0. Последняя фирма, кстати, передала Microsoft свою программу восстановления стертых файлов, полная версия которой входит в пакет PC Tools 7.0, получив в обмен право использовать в своем новом инструментальном пакете интерфейс, практически аналогичный оболочке MS-DOS Shell 5.0. Фирмы Novell и Banyan Systems сообщили, что их сетевое обеспечение также поддерживает MS-DOS 5.0.

По сведениям, поступающим из московского представительства корпорации Microsoft, завершена работа над русскоязычным исполнением новой версии операционной системы MS-DOS 5.0. В европейском отделении Microsoft в Ирландии уже приступили к производству этого нового программного продукта, в ближайшие месяцы он поступит в продажу в нашей стране. Оперативность Microsoft в изготовлении русскоязычной версии может вселить надежду, что беспредел нашего пиратского копирования программ пока не остановил сизифовых усилий этой могущественной корпорации по приобщению нас к самой передовой высокотехнологической цивилизации и обеспечению отечественных пользователей русскоязычными версиями своих новейших программных продуктов. Однако долго ли продлится такая бескорыстная благотворительность? Вовсе не нужно быть Нострадамусом, дабы с уверенностью предсказать, что вечной она не будет. А тем временем в зарубежную прессу уже проникают непроверенные сообщения, что Microsoft не остановилась на достигнутом и уже на всех парах разрабатывает следующую версию MS-DOS (шестую?), которая якобы будет принципиально отличаться от нынешней версии 5.0 и будто бы будет поддерживать новую файловую систему HPFS (High Performance File System операционной системы OS/2), а также интерфейсы DPMI (DOS Protected Mode Interface, реализованный в Windows 3.0 и в OS/2 2.0) и VCPI (Virtual Control Program Interface фирм Phar Lap и

Quarterdeck), а кроме того, будет работать в защищенном режиме.

Подтвердятся ли эти слухи и доведется ли нам когда-то увидеть следующую русскоязычную версию MS-DOS — покажет будущее. А сегодня русскоязычная версия MS-DOS 5.0, при создании которой была учтена справедливая критика, прозвучавшая в адрес предыдущей версии MS-DOS 4.01, наконец-то, готова, и мы искренне приветствуем ее появление на просторах нашей чудесной родины.

Что же нового в DOS 5.0? Кратко перечислить все небольшие изменения и улучшения едва ли возможно. В двух словах можно лишь сказать, что, хотя каких-то радикальных перемен не произошло, новая операционная система стала гораздо более удобной, более безопасной и более комфортабельной для пользователей-новичков. Некоторые "профи" могут, не без некоторого цинизма, утверждать, что новая DOS утратила былое лидерство и плетется в хвосте тех технологических достижений, которые значительно раньше удалось реализовать другим разработчикам программных продуктов. Однако в оправдание Microsoft (разумеется, если деятельность Microsoft вообще нуждается в какой-либо апологии) могу заметить, что любая новая версия DOS должна быть чуточку консервативной, отбирая и интегрируя в себе лишь самые лучшие и нужные новшества и не бросаясь из стороны в сторону в погоне за яркими и блестящими безделушками. Впрочем, уже то, что это новая операционная система, которой наверняка суждено серьезно изменить технологию управления миллионами персональных компьютеров, само по себе достаточно важная и злободневная новость. Поэтому лучше, наверное, все-таки начать с того, что в ней не изменилось, а соответствует нашим привычным представлениям об операционной системе DOS.

В этой версии MS-DOS в большой степени сохранена преемственность с предыдущими версиями. Сохранилась прежняя таблица распределения файлов (FAT). Имена каталогов по-прежнему могут состоять из одиннадцати символов ASCII, которые использовались в прошлом. Командная строка также может содержать максимум 126 символов. Оставлен без изменений и прежний архаичный командный язык пакетных BAT-файлов, о якобы предстоящем радикальном изменении которого распространялось немало слухов. Не подтвердилось и то, что в новой версии MS-DOS будет заменен файл COMMAND.COM по лицензии фирмы J.P. Software, которая широко известна своим популярным пакетом shareware-программ 4DOS, улучшающим и облегчающим работу пользователя в операционной системе DOS. Никаких заметных изменений не претерпела в MS-DOS 5.0 и поддержка последовательных периферийных устройств, ленточных накопителей, дисководов оптических дисков и принтеров. Впрочем, новинка все же есть: теперь поддерживаются дисководы для 3,5" флоппи-дисков емкостью 2,88 Мбайт.

Версия MS-DOS	Максимальный объем рабочей памяти	Максимальный объем памяти при работе в сети
3.3	575 Кбайт	490 Кбайт
4.01	565 Кбайт	480 Кбайт
5.0	621 Кбайт	613 Кбайт

Эти цифры приводятся Microsoft применительно к машинам на процессорах 386. В зависимости от конкретной конфигурации машины и используемого программного обеспечения возможны некоторые отклонения от указанных значений.

Одним из самых заметных положительных изменений в MS-DOS 5.0 можно считать новый метод размещения в памяти. Ядро новой DOS, то есть внутренняя резидентная часть операционной системы, управляющая процессами работы остальной операционной системы и распределяющая для них физические ресурсы машины, заметно сокращено в размерах в сравнении с MS-DOS 4.01. От этого, впрочем, было бы мало проку, если бы не еще одно новшество, оказавшееся самым сенсационным: во всех современных машинах и во многих старых (за исключением тех машин на процессоре 8088, которые не имеют расширенной памяти). Ядро DOS, для машин на процессоре 80286, может размещаться в области HMA (адреса выше 1 Мбайта) и, для машин на процессорах 80386 или 80486 — в областях HMA и UMB, (адреса от 640 Кбайт до 1 Мбайта), освобождая для использования прикладными программами рабочее пространство до 620 Кбайт. Именно это замечательное свойство MS-DOS 5.0 однозначно гарантирует новому программному продукту самое массовое признание и распространение во всем мире. Несомненно, дополнительное рабочее пространство памяти окажется очень кстати всем пользователям, которые используют такие "жадные до памяти" программы, как приложения для графической среды Windows 3.0, Ventura Publisher, FoxPro и многие другие прикладные программные продукты.

Кроме этого "памятного" момента, есть и другая хорошая новость, касающаяся распределения ресурсов дополнительной памяти. В MS-DOS 5.0 включено средство EMM386.EXE, аналогичное известной утилите QEMM-386 фирмы Quarterdeck, позволяющее пользователям машин с процессорами 386 и 486 с помощью команд DH (DEVICEHIGH) и LH (LOADHIGH) убирать драйверы устройств и резидентные программы из рабочей части оперативной памяти в "верхнюю" дополнительную EMS-память. Так что, если вы еще

не успели обзавестись пакетом DESQview-386, в который входит замечательная утилита QEMM-386, новая версия MS-DOS 5.0 будет вашим пропуском в прекрасный новый мир самых высоких технологий управления памятью. Впрочем, похоже, использование столь изощренных постмодернистских технологий пока по карману лишь очень немногим отечественным пользователям персональных компьютеров.

В оболочке DOS Shell, которой практически пользовались лишь очень немногие владельцы предыдущей MS-DOS 4.01, предпочитая более привычный Norton Commander, произошли весьма существенные перемены. Создатели новой оболочки явно постарались учесть и ликвидировать прежние недостатки, чтобы сделать работу в новой оболочке более комфортной и приятной. Новая DOS Shell может работать как в текстовом, так и в графическом режиме, а ее пользовательский интерфейс явно испытал облагораживающее влияние Windows. Вначале может показаться, что новая оболочка функционально весьма похожа на многочисленные программные продукты других фирм, выполняющих привычный круг задач по управлению файлами: просмотр иерархической структуры дерева каталогов и содержимого текстовых файлов, копирование, удаление и перемещение файлов с помощью мышки. Однако дальше начинаются различия: управление программами в новой DOS Shell аналогично принципам, используемым в Windows и в Presentation Manager OS/2. Вы можете не продираться сквозь раскидистое дерево каталогов, а оперировать лишь ограниченной группой необходимых прикладных программ. Несомненно, для большинства рядовых пользователей такое управление программами интуитивно более понятно и просто, чем освоение иерархической древовидной структуры хранения файлов на жестком и других дисках.

Появилась в новой оболочке и модная ныне возможность осуществления многозадачного режима работы: так называемый переключатель задач (Task Swapper — в англоязычной версии DOS 5.0) позволяет пользователю оболочки DOS 5.0 мгновенно переключаться из одной прикладной программы в другую, например, переходить из Lotus 1-2-3 в MS Word или в dBASE. Очень вероятно, что здесь компьютерные "профи" наверняка снобистски запротестуют: в MS-DOS 5.0 нет никакой подлинной многозадачности и эти уловки рассчитаны на наивных простаков. Действительно, в отличие, скажем, от DESQview, переключение задач в оболочке MS-DOS 5.0 основывается на переключении не в оперативной памяти, а только на диске. Переключатель задач лишь симулирует многозадачность, хотя никакого одновременного выполнения задач на самом деле не происходит. Поэтому с помощью такой псевдомногозадачности нельзя, скажем, одновременно выполняя

две программы, "вырезать" кусок изображения из одной прикладной программы и "вклеить" его в другую, как это возможно с помощью DESQview, Software Carousel или Windows. Однако, можно предположить, что большинству пользователей DOS настоящая многозадачность на самом деле не очень-то и нужна, тогда как реализованная в MS-DOS 5.0 рудиментарная псевдомногозадачность наверняка окажется удобной, полезной и продуктивной для многих неискушенных конечных пользователей. А профессионалы наверняка смогут использовать для реализации подлинной многозадачности другие программные средства.

Команды DOS также подверглись достаточно заметным изменениям. Появились некоторые новые команды, пополнился синтаксис многих старых команд. Помимо того, что в оболочке DOS имеется удобная справочная система, теперь и любая внешняя или внутренняя команда откликается на аргумент /? в своем синтаксисе, выводя на экран краткое описание. Многим, наверное, приходилось видеть приклеенные на мониторе бумажки, которые служат шпаргалками забывчивым пользователям и на которых нередко записывается синтаксис таких команд, как FORMAT или PRINT. Отныне для эффективной работы со всеми командами DOS совершенно не нужно листать справочное руководство: ответы на любые вопросы всегда под рукой. Разумеется, такое замечательное справочное новшество DOS можно только приветствовать, лишь удивляясь задним числом, почему же этого не было сделано раньше?

Команда FORMAT с аргументом /q теперь позволяет осуществлять быстрое форматирование дисков, которые ранее уже были отформатированы. Быстрое форматирование обнуляет таблицу распределения файлов и каталог, но данные при этом остаются не тронутыми. Это, во-первых, позволяет заметно экономить время при переформатировании дисков, которые находятся в хорошем состоянии, а, во-вторых, если пространство на диске позволяет, таблица FAT и корневой каталог сохраняется где-нибудь на диске, чтобы с помощью команды UNFORMAT вы смогли бы вновь при необходимости восстановить утраченную структуру. Более полная утилита UNFORMAT входит в пакет PC Tools 7.0, а в новой версии DOS она появилась по лицензии, предоставленной Microsoft фирмой Central Point Software.

Другая новинка, с которой также уже должны быть знакомы пользователи последних версий пакетов PC Tools и Norton Utilities — это резидентная программа MIRROR, предназначенная для улучшения условий безопасной сохранности данных на дисках. В специальном защищенном файле MIRROR.FIL постоянно отслеживается и сохраняется текущая файловая структура диска. Этот файл всегда присутствует в корневом каталоге жесткого диска и переписывается при каждой перезагрузке компьютера, а его предыдущая резервная копия сохраняется в файле MIRROR.BAK. Если в AUTOEXEC.BAT указана команда MIRROR C:, в файле MIRROR.FIL сохраняется копия FAT диска C:. Но можно сохранить в MIRROR.FIL также отраженные образы и других дисков, указав их поименно. Например, так:

```
MIRROR C: D: E:
```

Программа MIRROR может работать резидентно, постоянно отслеживая уничтожение файлов. Для этого достаточно выполнить команду MIRROR с аргументом /t. Файл MIRROR.FIL может быть затем использован сопутствующей утилитой UNDELETE для восстановления уничтоженных файлов, разумеется, если их место на диске не было уже использовано для записи других файлов. В отличие от ранних версий утилит типа UnErase из Norton Utilities, с помощью UNDELETE и MIRROR нет нужды вспоминать и подставлять стертую первую букву в восстанавливаемом имени файла, так как файловая структура полностью отражается в MIRROR.FIL. Точно так же с помощью упомянутой ранее утилиты UNFORMAT с диска читается скрытый файл MIRROR.FIL, если применялось быстрое форматирование по команде FORMAT с аргументом /q, и поэтому все случайно уничтоженные при форматировании дискет данные могут быть затем успешно восстановлены. А если MIRROR запускается с аргументом /partn, по которому сохраняется структура разбиения жесткого диска, с помощью утилиты UNFORMAT можно выкарабкаться из гораздо более серьезной переделки.

Во многих популярных утилитах и оболочках, применяемых сегодня пользователями персональных компьютеров, есть возможность просматривать содержание диска или каталога, подбирая файлы по определенным параметрам. Утилита DIR новой версии DOS может теперь воспроизводить на экране еще и перечень файлов, которые включаются в список или исключаются из него по желательным параметрам, а также сортировать файлы по имени, расширению, размеру или дате. Между прочим, имена файлов теперь воспроизводятся DIR не заглавными, а строчными буквами.

Кроме того, многие утилиты и оболочки, например Norton Commander, позволяют восстанавливать "историю" введенных команд и выполнять простые макрокоманды. Наконец-то, и DOS имеет такую возможность. Маленькая резидентная сервисная программа DOSKEY, впервые появившаяся в DOS 5.0 и занимающая в памяти всего 4 Кбайта, позволяет вывести из буфера памяти на экран список ранее введенных команд. Достаточно нажать функциональную клавишу F7, и вы получите на экране перечень всех команд, выполненных после включения компьютера. Используя клавиши управления курсором, можно затем выбрать нужную команду, частично отредактировать ее и выполнить снова. С помощью этой утилиты можно также выполнять

макрокоманды с различными изменяемыми параметрами по нажатию установленных клавиш. Имеется возможность присваивать макрокомандам любые удобные для запоминания названия, объединять их в цепочки, используя в качестве разделителя макрокоманд два символа — $T. Например, если вы хотите, чтобы по макрокоманде ДИСК форматировалась, а затем проверялась дискета в дисководе А:, можно написать такую макрокоманду:

```
DOSKEY ДИСК=FORMAT A: $T
    CHKDSK A:
```

Разумеется, допустимо писать не только заглавными, но и строчными буквами. Для более сложных макрокоманд в синтаксисе DOSKEY могут применяться пакетные параметры $1-$9, которые эквивалентны фиктивным параметрам %1-%9 в командных пакетных файлах. Например, если в вашем компьютере два дисковода для гибких дисков, предыдущую макрокоманду можно сделать более удобной, если написать:

```
DOSKEY ДИСК=FORMAT $1 $T
    CHKDSK $1
```

Если вам теперь понадобится отформатировать и проверить дискету в дисководе В:, достаточно написать в командной строке ДИСК В: и нажать клавишу Enter. А проявив еще немножко фантазии, можно создать гораздо более сложные и изощренные макрокоманды, которые значительно облегчат жизнь за клавиатурой компьютера.

Еще одна радостная новость: давным-давно морально устаревший строчный редактор текстов EDLIN наконец-то ушел на покой. Место этого музейного экспоната теперь занял новый и вполне современный полноэкранный текстовый редактор EDIT — наверняка он многим придется по вкусу. Причем EDIT.COM просто-напросто запускает текстовый редактор в файле QBASIC.EXE, который имеет непосредственное происхождение от известного QuickBASIC. И еще новость: QBASIC заменил в DOS 5.0 ушедший в отставку старый GW-BASIC. Впрочем, присутствие QBASIC.EXE — скорее ловкий рекламный трюк; он отнюдь не заменяет настоящий QuickBASIC, хотя и имеет аналогичную гипертекстовую справочную систему, отладчик и такие же приемы редактирования, но компилирует только в памяти. Впрочем, превосходный текстовый редактор QBASIC, который, кстати, удобно управляется мышкой, сполна оправдывает введение в DOS 5.0 этой демонстрационной программы.

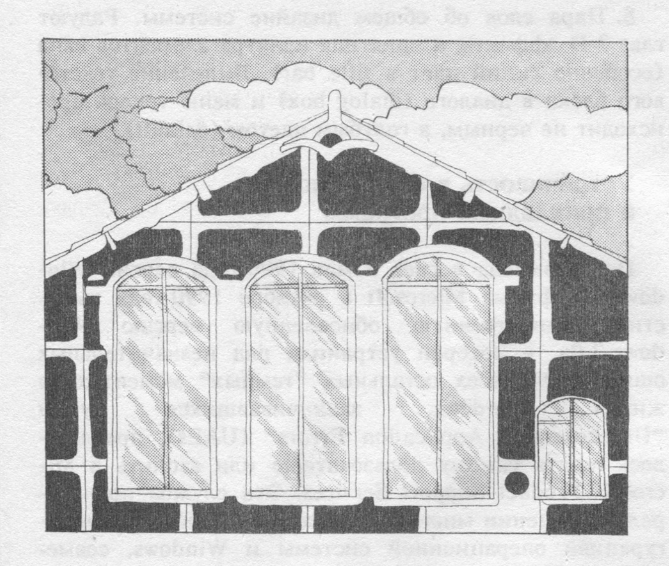
MS-DOS 5.0 поддерживает гораздо большее по объему прямое разбиение внешних накопителей: до 2 Гбайт вместо прежних 32 Мбайт. А пользователям особых разбиений диска, осуществляемых программами Disk Manager и SpeedStor, теперь предоставляется новая версия драйверов DMDRVR.BIN и SSTOR.SYS. С новой операционной системой могут использоваться прежние сетевые драйверы, но имеется и новый набор сетевых драйверов. Новое, полностью переработанное, руководство пользователя подробно объясняет приемы оптимальной настройки системы, а в текстовых файлах README содержится дополнительная информация о способах преодоления возможных проблем аппаратной и программной несовместимости. Новая версия DOS 5.0 предназначена для любых персональных компьютеров с процессорами от 8088 до 486, с объемом оперативной памяти не менее 512 Кбайт, и при установке на жестком диске занимает 2.8 Мбайта.

Выпуск новой версии MS-DOS — очень сложная и трудная задача. В процессе создания новой версии корпорация Microsoft более года работала в тесном контакте с семью тысячами пользователей и дистрибьюторов программных продуктов, чтобы "выловить всех блох" и в дальнейшем обеспечить всему миру быстрый безболезненный переход на новую операционную систему. Испытательные бета-версии MS-DOS 5.0 были известны многим отечественным специалистам уже с конца 1990 года. Похоже, что эти усилия не пропали даром. Все, кому часто приходится иметь дело с установкой операционных систем на новые машины, OEM (Original Equipment Manufacturer — изготовители комплексного оборудования, укомплектованного компонентами, изготовленными другими фирмами), многочисленные консультанты, эксперты и специалисты скоро смогут увидеть и испытать конечный результат этих трудов. Нет сомнения, что новая версия DOS немедленно превратится в фактический стандарт, с которым должны будут считаться все, вовлеченные в компьютерную индустрию.

По сведениям, полученным от московского представительства Microsoft, в первую очередь доступ к новой операционной системе получат отечественные OEM, которым будут предоставляться лицензии на производство русской MS-DOS 5.0 для комплектации выпускаемых ими персональных компьютеров. Однако русскоязычная MS-DOS 5.0 будет продаваться и отдельным пакетом для всех желающих, что обычно не практикуется в других странах. Ориентировочная цена пакета MS-DOS 5.0 будет составлять 100 долларов или 180 немецких марок. Наверняка, такой пакет можно будет купить и за соответствующее количество рублей. А сегодня Microsoft уже уведомляет своих дистрибьюторов о распродаже залежалых пакетов MS-DOS 4.01, которые спешно распродаются всего лишь за 90-95 марок. Microsoft также предлагает комплектный вариант распродажи MS-DOS 4.01 вместе с русскоязычной версией MS Works 2.0 за 210 немецких марок, тогда как рекомендованная базовая цена одного лишь пакета MS Works 2.0 первоначально составляла в 295 немецких марок.

А.Петроченков

214000, Смоленск, а/я 44
Телефон: (08100) 5-58-05



Что же новенького приготовил нам дядюшка Билл со своей компанией суперразработчиков? Вам интересно? Признаемся по секрету — нам тоже. Итак, вдоволь налюбовавшись пейзажем за тремя парадными окнами, попробуем заглянуть в то новое маленькое окошко, которое, хотя еще и прикрыто от любопытных взглядов, но уже вот-вот откроется. Давайте подойдем к нему потихоньку и заглянем в мир Windows...

Windows 3.1 Что новенького?

Предлагаемая здесь информация об MS-Windows 3.1 собрана из многочисленных журнальных статей, новостей USENET и дополнена или проверена автором на первой бета-версии продукта (май 1991 года).

При создании новой версии Windows 3.1 фирма Microsoft выделила следующие направления функционального развития продукта:

- увеличение производительности;
- устойчивость к сбоям системы и прикладных программ;
- включение технологии TrueType масштабируемых фонтов;
- увеличение интеграции программ;
- расширение системы для новых компьютерных платформ.

Увеличение производительности

Для увеличения производительности фирмой были предприняты следующие шаги.

1. Улучшен процесс установки системы на диск. Теперь SETUP различает и определяет большее количество типов "железа", конфигураций и TSR-ов, чем раньше, и позволяет пользователю почти не заботиться на этот счет. Хотя для пользователя процесс установки не изменился и об этих улучшениях он может только догадываться. В моей версии продукта были и неприятные моменты: программа "зависала" при работе из MS-DOS 5.0 (не бета) в режиме DOS = UMB, список предложенных драйверов мыши и видеоадаптеров был весьма скуден по сравнению с Windows 3.0a, но это мелочи.

2. Полностью переписан File Manager, и, поверьте, сделано это превосходно. Интерфейс программы очень похож на File Manager в Norton Desktop и позволяет создавать отдельные окна с "деревом" и списком файлов в директории. При переходе в другую директорию новое окно не создается, а обновляется текущее и весьма быстро. File Manager может запомнить состояние своих окон и полностью его восстановить при следующем запуске программы. В меню **Options** появилась настройка **Fonts** для установки фонта, которым идет отображение имен файлов и директорий, а форматирование дискеты может идти в режиме **Quick Format**. Есть еще одно объявленное, но, видимо, пока не реализованное свойство File Manager, касающееся более интуитивной модели манипулирования файлами. Например, для печати файла достаточно переместить иконку файла (drag and drop) из окна File Manager на иконку Print Manager. Пользователь будет

иметь возможность переместить иконку какого-нибудь файла на заголовок (title bar) работающей программы или на ее иконку, и этот файл будет загружен в программу. После работы с File Manager остается прочное желание постоянно его использовать.

3. **Program Manager** не претерпел таких серьезных изменений, как File Manager, хотя получил несколько долгожданных улучшений. Новая "обязательная" группа **STARTUP** запускает при загрузке Program Manager включенные в нее программы (если Program Manager работает как командный процессор — shell Windows). Строка описания программы в Program Manager (program item) автоматически делится на несколько строк. В команде **Run** наконец-то появилась опция **browse**, а опция сохранения состояния Windows перекочевала в меню программы.

4. **Print Manager** умеет теперь самостоятельно восстанавливаться после сбоев и продолжать работу. Например, когда у вас кончается бумага, печать возобновляется автоматически после приведения принтера в состояние **on-line**. Другим, еще не реализованным, улучшением будет появление универсального принтерного драйвера (UNIDRV), который позволит фирмам — производителям принтеров очень просто и быстро описывать параметры своих принтеров, используя небольшую таблицу. Около 250 принтеров будут поддерживаться в Windows 3.1 через универсальный драйвер.

5. В **Control Panel** изменению подверглась только настройка **Desktop**. Во-первых, благодаря опции выравнивания заголовков (Wrap title) имена программ под иконками могут выравниваться в две и более строки, не перекрываясь с именами других программ. Во-вторых, появился встроенный **Screen saver** с четырьмя модулями вариантов эффектов с разнообразной настройкой и опцией пароля.

6. Улучшена поддержка сети. Проблемы, возникающие при работе с сетью, легче определять и решать с помощью расширенной диагностики типа и источника проблемы. Другим важным изменением будет предоставленная пользователю возможность определить "постоянное" (persistent) соединение. При этом информация об удаленном принтере или диске сохранится в среде Windows после окончания сессии. Любое такое неподсоединенное устройство будет высвечиваться как недоступное в File Manager. Для возобновления соединения пользователю достаточно просто нажать на иконку диска мышью.

7. Улучшение производительности системы видно на глаз (не на самых мощных компьютерах). Увеличены производительность драйвера дисплея, скорость печати. Реализован быстрый страничный обмен (paging) в 386 расширенном (enhanced) режиме. Программы, вызывающие команду печати, быстрее получают обратно контроль для продолжения нормальной работы. Система будет включать в себя FastDisk — 32-разрядный драйвер, позволяющий Windows в обход DOS обратиться к BIOS для страничного обмена (paging) с файлом виртуальной памяти.

8. Пара слов об общем дизайне системы. Радуют глаз 3-D эффекты и приятная палитра элементов окна (особенно синий цвет в title bar). Выделение текстового блока в диалоге (dialog box) и меню теперь происходит не черным, а голубым цветом (default).

Устойчивость к сбоям системы и прикладных программ

После выхода в свет в мае 1990 года версии Windows 3.0 фирма Microsoft в октябре 1990 года выпустила единственную обновленную версию Windows 3.0a, в которой устранила ряд незначительных ошибок. Обо всех остальных "темных" моментах из жизни Windows, заканчивавшихся сухим "Unrecoverable Application Errors" (UAEs), предлагалось гадать самому пользователю или звонить в Microsoft Product Support Services. Эта служба поднаторела в решении многих вопросов, связанных с конфигурацией операционной системы и Windows, совместимостью TSR-ов и драйверов и т.д. Естественно, что всех проблем версия 3.0a не решила, поэтому в Windows 3.1 был сделан упор на следующие направления:

- улучшение диагностики события, вызвавшего ошибку;
- создание специальных средств в помощь разработчикам для написания программ, лишенных ошибок;
- изящное управление ошибкой в прикладной программе (ошибка не влечет за собой крах системы).

1. Диагностика ошибок сейчас происходит следующим образом: появляется сообщение о том, что ошибка случилась в такой-то программе, в таком-то модуле, по такому-то смещению. Такая диагностика действительно сильно упрощает процесс устранения ошибки. Дополнительно к этому версия 3.1 будет продаваться с программой "Dr. Watson", которая записывает информацию о случившейся ошибке. Данные этой программы помогут сделать "откат" от ошибки — проследить причины ее возникновения — а затем выявить и устранить ее.

2. К версии 3.1 фирма Microsoft выпустит средство для разработки надежных программ, устойчивых к ошибкам. Например, в систему будет включен новый механизм, позволяющий проверять многие параметры, через которые прикладная программа "общается" со средой Windows. Если программа использует неправильный тип параметров или их величина выходит за допустимый диапазон, то будет выдано сообщение об ошибке. В средства разработки будут включены некоторые новые утилиты определения источника проблемы. Например, новый "стресс тест" создает высокоактивную и динамичную среду, в которой ошибки разрабатываемой программы буквально "посыпятся" на программиста.

3. Если выполнение прикладной программы закончилось "зависанием", пользователь может воспользоваться клавишами Ctrl+Alt+Del, и Windows задаст вопрос о продолжении или прекращении выполнения

программы. Если пользователь ответит "прекратить", то прикладная программа завершится, Windows переустановит среду в стабильное состояние, что позволит продолжать нормальную работу других программ без перезагрузки всей среды Windows.

Технология TrueType масштабируемых фонтов

В Windows 3.1 включена новая технология масштабируемых фонтов TrueType. Она поддерживает контурное (outline) изображение фонтов и позволяет пользователю получать фонты любого размера. Обеспечивается высококачественный вывод на любой монитор или принтер, поддерживаемый Windows.

TrueType-технология является составной частью Windows 3.1. Пользователю не нужно ее дополнительно покупать или устанавливать, и любая программа может воспользоваться этими фонтами. В версию 3.1 будет включено четыре семейства (family) фонтов технологии TrueType: Arial (как альтернатива Helvetica), Times New Roman, Courier и Symbol.

Так как технология TrueType перенесена в Windows из Apple Macintosh, Microsoft позаботилась о том, чтобы фонты этой технологии могли без каких-либо изменений использоваться и в Windows, и на Macintosh.

Фонты TrueType автоматически конвертируются в растровое изображение (bitmap images) и загружаются в лазерный принтер. На экране пользователь видит то же самое, что и на распечатанной странице. TrueType использует динамическую загрузку фонтов в принтер, посылая не все символы, а только требующиеся, что повышает скорость и эффективность печати.

Несколько слов о вставших передо мной проблемах при использовании TrueType-фонтов. Во-первых, каждый тип фонта (regular, bold, italic, bold italic) имеет размер около 56 Кбайт, и загрузка фонтов в память обходится не дешево (по отношению к памяти). Во-вторых, Word for Windows, настроенный на Laser Jet IIIP (т.е. использующий принтерные фонты), не распознал и не вывел в список ни одного фонта TrueType. Хотя, если написать имя фонта вручную, все будет корректно работать вплоть до того момента, как вы захотите восстановить запомненный документ, оформленный фонтами TrueType. Я получил чистый лист на экране и периодическое "зависание" машины. В-третьих, были проблемы с русификацией, связанные с тем, что фонт Courier теперь TrueType, а Helvetica и Times Roman вообще отсутствуют. Мне пришлось отказаться от Courier Cyrillic и Small (еще один новый фонт preview из WFW или Excel).

Увеличение интеграции программ

1. Технология связи и внедрения объектов (Object Linking and Embedding).

Эта технология создает среду, в которой прикладная программа может различать (share) информацию об отдельных "кусках" своей среды. Через технологию OLE данные в зависимости от типа разделяются на объекты. Таблица, график (chart) электронной таблицы, параграф текста — все это примеры таких объектов. OLE обеспечивает возможность программам различать эти объекты очень легко.

Среда Windows 3.1 поддерживает OLE посредством стандартной библиотеки, интерфейса и протокола обмена объектами данных. Разработчики в среде Windows будут включать свойства OLE в программы, и пользователи получат новый уровень интеграции программ.

Свойства OLE сейчас включены в новые версии программ Windows Write, Paintbrush и Cardfile. Пользователь может, например, создать картинку в Paintbrush и перенести ее в Cardfile. Для исправления картинки достаточно два раза нажать на нее мышью (double click). При этом автоматически загрузится Paintbrush с картинкой. Вместо команды Save в Paintbrush появится команда Update, завершающая процесс вносимых изменений. Посредством технологии OLE пользователь получит высокоинтегрированную среду, где множество программ будут общедоступными инструментами для создания всевозможных объектов.

Правда, есть здесь некоторые "подводные камни": например, размер картинки, загруженной в Paintbrush из Cardfile, изменять уже нельзя. Впрочем, этого и следовало ожидать. В Cardfile также с успехом загружается таблица Excel 3.0 (в котором, кстати, протокол OLE реализован для внутренних целей) и с той же проблемой неизменяемости размера объекта. Тем не менее, технология OLE — одно из самых важных улучшений среды Windows.

2. Улучшение поддержки динамического обмена данными (Dinamic Data Exchange).

DDE является стандартным средством среды Windows для разделения (share) данных между программами. В версии 3.1 реализована новая Библиотека Управления Динамическим Обменом Данными (DDEML — DDE Manager Library), которая предлагает высокоуровневую модель программирования и упрощает процесс разработки свойств DDE в программах для Windows.

3. Улучшение поддержки DOS-программ.

Производительность DOS-программ увеличена, особенно когда Windows 3.1 работает вместе с MS-DOS 5.0 (так как MS-DOS 5.0 оставляет больше памяти для работы DOS-программ). Теперь также есть возможность запускать в окне DOS-программу, работающую в графическом режиме VGA. Версия содержит больше описаний PIF для существующих программ. Наконец, страничная подкачка (disk-paging) позволит пользователю запускать больше DOS-программ, чем под Windows 3.0.

Расширение системы для новых компьютерных платформ

Windows 3.1 будет иметь специализированные расширения для работы на новых компьютерных платформах.

1. Windows для Pen-компьютеров (Pen-based Computers).

Графический пользовательский интерфейс, хороший распознаватель символов и перо служат основой высокоинтуитивного и действительно "персонального" пользовательского интерфейса. Для использования потенциала Pen-компьютеров фирма Microsoft разработала серию расширений для среды Windows:
- возможность ввода информации с помощью пера;
- интерпретатор сообщений от пера, позволяющий существующим программам для Windows и DOS использовать перо;
- модуль, открывающий доступ к технологии распознавания символов.

2. Multimedia.

Расширения среды Windows 3.1 для multimedia позволят пользователю включать в существующие программы новые объекты — звук (audio), мультипликацию (animation), видео (full-motion video). Эти свойства открывают целый класс таких multimedia-документов, как энциклопедии, дополненные видео- и аудиоклипами, или каталоги, которые показывают движущиеся иллюстрации. Расширяемая архитектура Windows дает возможность компьютерам multimedia создавать дешевые системы для образования и для дома.

3. Поддержка Laptop.

Пользователи Laptop в версии 3.1 получат новое свойство, называемое "mouse blur", которое позволит легко находить курсор на дисплее Laptop. Фирма Microsoft будет продавать лицензии фирмам-изготовителям на версии Windows в ПЗУ (ROM). Это откроет путь к использованию новых типов компьютеров.

Штрихи к портрету

В версии Windows 3.1 появилась возможность применения системного подхода к написанию общеупотребительных диалогов типа: "Open..." и "Save as...". Эта возможность реализована через стандартную библиотеку COMMDLG.DLL. Это позволит вам иметь стандартный механизм выбора файлов, директорий, дисков и описаний масок файлов при работе ваших программ. В библиотеку также включена возможность использовать шаблон для диалога "About ...". Фирма предлагает включать в него имя и организацию лица, зарегистрировавшего пакет, количество свободной памяти, памяти, доступной среде, и режим работы Windows.

В настройке фонтов появилась возможность использовать псевдонимы. Так как многие существующие программы используют фонты Helvetica и Times Rmn, а в версии 3.1 их больше нет, то с помощью псевдонимов вы можете определить любой из имеющихся у вас фонтов как Helvetica или Times Rmn. По умолчанию они заменяются MS Sans Serif и MS Serif соответственно.

В середине сентября 1991 года фирма Microsoft выпустила вторую бета-версию продукта, а окончательный вариант появится к марту 1992 года. По информации, полученной в московском отделении фирмы, Интернациональная версия Windows 3.1 будет включать в себя встроенные средства поддержки кириллицы, украинского и белорусского языка. Эта версия появится через два-три месяца после выхода основного варианта продукта.

Было сообщено также, что между фирмами — производителями фонтов и русификаторов для среды Windows достигнуто соглашение на стандарт расположения кириллицы в таблице ASCII: со 192-"А" по 255-"я" символ.

А.Зеленков

По материалам:
USENET news from George MOORE (Microsoft Corp.) and Tom Haapanen (University of Waterloo).
PC Week.
Info World.

В предыдущем номере мы рассказали о компьютерах-блокнотах. Они делают работу очень динамичной, не сковывая ваших перемещений. Но часто приходится результаты этой работы представлять на бумаге. Здесь вам поможет

Citizen PN48 — принтер для компьютера-блокнота

Осенью 1991 года фирма Citizen выпустила в продажу новый принтер PN48 Notebook Printer. Впервые показанный в конце мая на выставке COMDEX, проходившей в Калифорнии, он ознаменовал новый этап в развитии технологии портативных устройств печати. Это первый портативный принтер, обладающий качеством печати, которому может позавидовать огромный настольный лазерный принтер. Раньше приходилось мириться с не слишком высоким качеством принтера, который можно было носить с собой.

Citizen PN48 — это машина, весящая всего лишь 900 г (1170 г с аккумуляторами) и печатающая 80 символов в секунду с разрешающей способностью 360 точек на дюйм. Размеры принтера также очень малы: высота 5 см, площадь основания 9x29 см. Теперь бизнесмен может иметь офис в портфеле с еще меньшими проблемами — габариты и вес делают PN-48 отличным дополнением к компьютеру-блокноту.

Принтер позволяет печатать на самых разных типах бумаги, в том числе на конвертах, этикетках и бланках. Кроме того, возможен вывод на пленку для проекции с помощью эпидиаскопа. Обычное деловое письмо печатается менее чем за минуту.

В NP-48 использована технология термопереноса — специальная лента при нагревании оставляет краситель на прижатой к ней бумаге. (Та же технология используется во всех современных цветных принтерах, дающих качество, сравнимое разве что с качеством превосходной цветной фотографии.) Возможны два режима печати: в первом из них, предназначенном для получения высокого качества печати, каждый участок ленты используется единожды. Второй режим — для печати черновиков. В нем каждый участок используется неоднократно, что повышает ресурс красящей ленты в пять раз. Одной кассеты в нормальном режиме хватает на вывод 35,000 символов, упаковка из шести кассет стоит 15 долларов. В итоге стоимость печати получается не слишком высокой, хотя страница обходится дороже, чем при использовании лазерного принтера, но вы же не собираетесь держать этот принтер в качестве основного. Да и за портативность, конечно, приходится платить.

В принтере есть встроенный шрифт Courier, расширенный дополнительными эффектами вроде теней, обводного шрифта и так далее. Разумеется, принтер оснащен автозагрузкой бумаги.

Принтер питается от встроенных никель-кадмиевых аккумуляторов, весящих всего 200 г. Заряда батарей хватает примерно на четверть сотни страниц, что вполне достаточно для распечатки контракта нормального размера. Цена принтера — 549 долларов, но уже в ноябре в Штатах его можно было купить за 370.

К принтеру, кроме всего прочего, прилагается руководство, выполненное в виде гипертекстовой системы на одной 3.5-дюймовой дискете.

Принтер совместим с большим количеством популярных пакетов, а при работе с Windows позволяет использовать все его возможности по работе со шрифтами.

И.Вязаничев

Использованы материалы:
Newsbytes News Network, May 20, 1991.
Publish, September 1991.

Агентство "КомпьютерПресс"

ЗАКАЗ НА ФЛОППИ-ЖУРНАЛ

Ежемесячные обзоры материалов компьютерных журналов Америки, Европы и Японии по актуальным проблемам применения современной вычислительной техники и программного обеспечения, публикуемые на страницах компьютерного сборника "КомпьютерПресс", пользуются большой популярностью среди отечественных пользователей ПЭВМ. Однако объем сборника не позволяет в полной мере удовлетворить интерес специалистов к некоторым достаточно узким, но актуальным проблемам компьютеризации. Поэтому редакция сборника, рассчитанного на широкую аудиторию пользователей ПК с самыми разнообразными запросами, считает необходимым приступить к изданию ряда специализированных приложений. Первое из них:

ФЛОППИ-ЖУРНАЛ "БАЗЫ ДАННЫХ И ЭКСПЕРТНЫЕ СИСТЕМЫ"

В нем Вы можете найти:

- аналитические обзоры материалов ведущих компьютерных журналов Америки, Японии и Европы, а также авторские статьи по проблемам разработки перспективных и применения уже существующих систем управления базами данных (СУБД) и экспертных систем (ЭС);
- анализ состояния и тенденции национальных и международного рынка программной продукции;
- практические рекомендации пользователям по работе с конкретными системами;
- информацию о зарубежных и отечественных фирмах и разработчиках СУБД и ЭС;
- календарь национальных и международных выставок, конференций и семинаров;
- экспресс-новости и рекламу.

Кроме того, в журнале будут публиковаться аннотации статей ведущих отечественных компьютерных журналов.

И, наконец, под эгидой журнала будут проводиться ежегодные некоммерческие конференции по проблемам систем управления базами данных и экспертных систем (участие докладчиков оплачивается журналом).

Журнал будет выходить на дискетах (по одной дискете 5.25"x360 Кбайт в каждом выпуске) шесть раз в год и распространяться только по подписке. Стоимость подписки на год (6 выпусков) — 450 руб., на полгода (3 выпуска) — 225 руб.

Для того, чтобы подписаться на флоппи-журнал "Базы данных и экспертные системы", достаточно перечислить соответствующую сумму (450 или 225 руб.) на расчетный счет No. 1467846 в Коопбанке Центросоюза СССР, корреспондентский счет No. 161406 в ЦОУ Госбанка СССР МФО 299112, заполнить прилагаемый бланк ззказа и отправить его вместе с копией платежного документа по адресу: 123459 Москва а/я 20.

Контактный телефон в г. Москва — (095) 496-0578.

ПЕРВЫЙ ВЫПУСК ФЛОППИ-ЖУРНАЛА "БАЗЫ ДАННЫХ И ЭКСПЕРТНЫЕ СИСТЕМЫ" — В МАРТЕ 1992 г.

ПРИГЛАШАЕМ ВАС К СОТРУДНИЧЕСТВУ С ЖУРНАЛОМ НА ВЗАИМОВЫГОДНОЙ ОСНОВЕ

ПОДПИСАТЬСЯ НА НАШ ЖУРНАЛ НИКОГДА НЕ ПОЗДНО



В этой статье читатель найдет полезные сведения об одной из разновидностей коммерческого программного обеспечения — программ проверки текстов на соответствие правилам русской орфографии, довольно быстро завоевавшей популярность у пользователей. Авторы статьи уже несколько лет работают над одной из популярнейших систем этого класса — программой ОРФО.

Как выбрать программу проверки орфографии

На отечественном рынке первые программы проверки правописания появились немногим более полутора лет тому назад. К этому времени вкусы потенциальных покупателей были уже сформированы отчасти рекламой, отчасти привычкой работать с аналогичными системами для английских текстов.

В представлении покупателя, такая программа должна уметь достаточно быстро проверять текст, исправлять ошибки, предлагать варианты правильного написания слов, в которых были допущены ошибки, обладать возможностью пополнения словаря.

Однако пользователь должен знать, что программа проверки русской орфографии, может содержать и незаметные при покупке недостатки, которые в дальнейшем сделают ее использование неудобным и негодным. Здесь мы расскажем, опираясь на свой опыт и отзывы покупателей, какие детали требуют от покупателя особенно пристального внимания.

Несколько слов о программе ОРФО. Это резидентная программа, позволяющая использовать для проверки и правки текста текстовый режим любого текстового процессора. Программа проверяет текст, видимый в окне текстового процессора, и подсвечивает орфографические ошибки в словах, ошибки в согласовании прилагательных с существительными по роду, числу и падежу, ошибочно набитые вместо русских букв латинские, а также правильность расстановки знаков препинания и употребления прописных и строчных букв.

Практика показывает, что наиболее требовательными пользователями программы проверки орфографии являются редакторы издательств, имеющие дело с большими объемами текстов. Другие придирчивые пользователи — иностранцы, пишущие на русском, особенно преподаватели русского языка за рубежом. Здесь мы обсудим замечания именно этих категорий покупателей.

Итак, что же должна уметь программа проверки правописания:

1. Проверка текста

По мнению большинства пользователей, проверка орфографии должна производиться во всем текстовом файле. При этом программа должна давать возможность вернуться к ранее найденным ошибкам, а также изменить текст вокруг ошибки (контекст), а затем продолжить проверку. Большинство известных нам

программ проверки русской орфографии не предоставляют пользователю такой свободы при проверке и правке текста, заставляя его проверять файл только сверху вниз, и позволяя править только само ошибочное слово (как, впрочем, и большинство программ проверки английской орфографии).

Учитывая пожелания пользователей ОРФО, в ОРФО 2.0 была включена функция проверки всего текстового файла, довольно неожиданная для резидентной программы. Автоматически пролистывая файл на экране и проверяя его до первой ошибки, ОРФО подсвечивает сразу все ошибки, попавшие на экран, и позволяет править любую из них или с помощью средств ОРФО, или средствами используемого текстового процессора, либо сразу же перейти к проверке следующей страницы.

Во вторую версию ОРФО, также с учетом предложений пользователей, добавлен режим автоматической проверки, благодаря которому при вводе текста с ошибкой сразу раздается звуковой сигнал.

2. Программа должна быть совместимой с рабочей средой

При покупке нужно обязательно удостовериться, что программа работает на вашем типе компьютера, с вашим типом монитора, и с кириллицей в используемой вами кодировке.

Владельцы монохромных мониторов и нестандартных кодировок кириллицы знают, сколь многие программы оказываются для них неудобны или бесполезны.

Сходную проблему представляет собой обработка различных текстовых форматов, т.е. возможность проверять тексты, созданные с помощью популярных текстовых процессоров, например, Microsoft Word и Wordperfect.

Именно желанием разработчиков раз и навсегда обеспечить такую совместимость и объясняется, что ОРФО — резидентная программа, работающая с любым текстовым процессором. Для проверки и коррекции текста с помощью пролистывания его в окне текстового процессора неважно, какой формат имел этот текстовый файл до его вызова на экран, это касается только самого текстового процессора.

Конечно, такой прием позволяет обрабатывать только тексты, созданные в текстовом режиме, и исключает работу с текстовыми процессорами и издательскими системами, использующими только графический режим экрана (подобными Chiwriter, Ventura Publisher и всем текстовым процессорам под Windows).

По сведениям авторов, файлы этих форматов до сих пор не обрабатывает вообще ни одна известная программа контроля русской орфографии.

Иногда за техническими и системными чертами программы бывает непросто увидеть и оценить ее интеллектуальное наполнение.

3. Лингвистический интеллект программы — как его увидеть?

Разработчики программ контроля правописания любят поражать воображение покупателей тем, сколько тысяч слов в словаре их программы. Однако на самом деле эти цифры почти ничего не означают. Действительно, что вам скажет тот факт, что словарь ОРФО 2.0 содержит более 200,000 основ? Ведь в словаре находятся основы слов, к которым при анализе текста программа приписывает окончания и фрагменты чередования, а сколько слов программа может таким образом узнать и проверить в реальном тексте, понять очень трудно. Это зависит от количества и качества правил образования слов из основ в данной лингвистической модели.

Например, для основы глагола “делать“ одна программа может узнавать и проверять в тексте все его глагольные формы — “делаю“, “делаешь“, “делаете“ и так далее — всего около двадцати, а другая программа может еще узнавать и все формы его четырех причастий — “делающий“, “делаемый“, “делавший“, “деланный“ во всех их падежах, родах и числах, — еще около восьмидесяти однокоренных слов, узнаваемых в тексте. Ясно, что словари этих программ могут содержать одинаковое количество основ, но при этом, вторая программа будет проверять текст точнее.

Применение правил образования слов с дефисом и приставками также может сильно увеличивать количество слов, узнаваемых программой в тексте. Например, если в словаре системы есть слова вроде “вызвал умную программу“, полезно выяснить, узнает ли программа контроля орфографии слова “вызванная-таки, вызывал-вызывал, умный-умный, умную-преумную, поумнее, полпрограммы, программу-то, программе-корректору“ и так далее.

Покупателю кажется естественным определить качество словаря программы именно по этому критерию — количеству не узнанных ею правильных слов в обычном тексте. Для этого обычно покупатели используют свои собственные проверочные тексты. При проверке такого текста правильно написанные, но не известные программе слова сразу бросаются в глаза.

Пропущенные же ошибки невидимы и учету поддаются плохо. Однако количество пропускаемых ошибок — еще один очень важный критерий интеллекта программы.

Если разработчики программы не утруждали себя точным описанием индивидуального способа изменения каждого слова и заложили в нее слишком простые и вольные правила словообразования причастий, множественного числа существительных, кратких форм прилагательных, слов с дефисом и приставками, то может случиться, что программа разрешит писать “прыгаемый“, “мужественностями“, “дворцовы“, “смазочна“, “стол-таки“, “пользователя-новичку“, “преумный“ и тому подобное, пропуская настоящие ошибки в тексте (особенно недописанные или неис-

правленные окончания слов). Для изучающего русский язык такой покладистый корректор окажется вообще бесполезным, так как он пропустит именно характерные ошибки.

Увидеть, насколько строго программа проверяет правила образования слов (т.е. оценить ее нормативную функцию) можно с помощью текста из "плохих" слов и характерных ошибок, однако пользователю такой тест довольно трудно составить самому.

Гораздо проще увидеть недочеты нормативной функции исследуемой программы с помощью наблюдения за поиском замены к ошибочному слову. Попросите программу найти замену какому-нибудь короткому слову и оцените количество бессмысленных и некорректных вариантов, выдаваемых программой в качестве возможной замены. При проверке текста все эти ошибочные слова будут пропущены программой как правильные.

Совершенствование нормативной функции программы — трудный путь, он требует многих человеколет работы (например, на морфологическую модель и словарь ОРФО 2.0 затрачено более 10 высококвалифицированных человеко-лет).

4. Набор полезных умений и функций

При оценке лингвистического интеллекта программы следует также обратить внимание на то, умеет ли исследуемая программа выполнять следующие функции:

Проверка правописания слов, содержащих большие буквы

Знает ли тестируемая программа, что слова "Москва", "Иван" пишутся только с большой буквы, или она разрешит написать их и с маленькой (конечно, для этого требуется, как минимум, наличие в словаре программы собственных имен и названий).

Например, программа ОРФО обнаружит в тексте неверно написанные слова "Санкт-петербург" и "Рио-Де-Жанейро" и предложит правильные варианты ("Санкт-Петербург", "Рио-де-Жанейро") на замену. Это свойство программы проверки правописания особенно полезно при проверке текстов, содержащих много фамилий и имен.

Проверка правописания сложных слов с дефисом

Слова с дефисом — камень преткновения для многих иностранцев и русских. Как нужно правильно писать — "естественно-научный" или "естественнонаучный", "полуживой" или "полу-живой", "западно-сибирский" или "западносибирский"?

Очевидно, если программа такова, что разрешает написать через дефис любые два правильных слова, то слово "естественно-научный" будет считаться написанным правильно (а оно неправильное).

ОРФО 2.0 в этом случае обязательно найдет ошибку и поставит соответствующий диагноз. Пользователь увидит на экране сообщение "это слово обычно пишется слитно".

Проверка сокращений на наличие точки за ними

Эта возможность обязательно нужна для проверки отчетов, форм, анкет, содержащих много принятых в них сокращений: "преп., ст., корп., акад., ред., ген., корр." и так далее.

Обнаружение малой буквы в начале предложения

Такая ошибка — случайная маленькая буква после точки — встречается довольно часто.

ОРФО 2.0 разбирает каждую точку в тексте и контекст вокруг нее на соответствие одной из нескольких десятков схем (точка после сокращения, отточие, точка до кавычки и после, точка после инициалов и так далее), и решает в каждом случае, допустима ли здесь малая буква в следующем слове.

Проверка знаков препинания на соответствие корректорским требованиям

Кроме случайной малой буквы после точки существует большое количество довольно частых ошибок в знаках препинания. Сюда относятся и просто напечатанные дважды запятые и точки, и другие знаки препинания, и отсутствие пробела между запятой и следующим словом, и неправильно перенесенные программой форматирования на другую строчку (например, оторвавшийся от предложения вопросительный или восклицательный знак) и многие другие ошибки. ОРФО 2.0 проверяет сочетания знаков препинания на соответствие более чем ста правилам сочетаемости и расположения в тексте.

Обнаружение и исправление вкравшихся в слово латинских букв или цифр. Каждое русское слово с нечаянно набитой латинской буквой программа должна давать возможность автоматически исправить, так как часто латинская буква на глаз не отличается от русской, а на печати может иметь небольшие отличия.

Проверка слов с приклеившимися к ним цифрами — дело не простое, так как программа не должна бы подсвечивать в технических текстах названия микросхем, транзисторов и так далее.

К сказанному добавим, что все без исключения иностранцы, пишущие на таком трудном русском языке, высоко оценивают возможность ОРФО посмотреть на экране все формы любого правильного слова, хотя эта функция и не служит непосредственно проверке текста.

Пользователь программы проверки орфографии должен также иметь возможность посмотреть подробные пояснения по каждому трудному случаю правописания.

Многие редакторы издательств отмечают также, что хотя бы простая проверка согласования слов и синтаксиса в предложении жизненно важна для них. Они готовы мириться с подсветкой мнимых ошибок (неполных предложений, разговорной речи и т.п.)

ради обнаружения как можно большего числа настоящих ошибок.

И действительно, функция проверки согласованности слов в предложении дает возможность находить даже ошибки в правописании слов, которые не удается найти, проверяя просто отдельные слова. Например, ни одна программа, проверяющая только правописание слов, не найдет ошибку в предложении "Длинна моста сто метров", ибо слово "длинна" правильное — оно есть в словаре как краткое прилагательное. Однако проверка согласованности слов сразу обнаружит, что краткая форма прилагательного "длинный" здесь неуместна, так как не имеет соответствующего согласованного существительного.

Ошибки всех перечисленных типов программа должна обнаруживать также и в новых словах, введенных пользователем.

Для исправления ошибки любого типа очень полезно видеть на экране соответствующий диагноз программы.

Удобной чертой ОРФО 2.0 является именно такая способность давать развернутые пояснения к подсвеченным ошибкам. Для каждой из них пользователь может посмотреть краткий диагноз, а также подробное объяснение причин, почему данная ошибка подсвечена.

При выборе программы контроля правописания нужно внимательно изучить средства пополнения словаря: очень важно, как вводятся в словарь новые слова — только в том виде, что в тексте (графически) или сразу во всех формах, с возможно более полной информацией. Придется ли вам последовательно вводить все формы вашей фамилии и затем обнаружить, что ту же фамилию, но в женском роде (фамилию вашей жены) программа все равно не узнает?

Что касается ОРФО, то ее лингвистические знания позволяют определить правописание всех форм слова (которых бывает до 20) после трех-четырех вопросов к пользователю. Идеальная программа, по нашему мнению, должна определять все формы нового слова, задавая один-два вопроса, а для некоторых слов — ни одного.

5. Дружелюбный интерфейс

Привычка к нелицензионным (т.е. краденым) западным программам сделала свое дело образования массы пользователей. Многие из них уже понимают, что программа должна быть:

а). Красивой. Это ясно всем, кроме многих советских программистов, до сих пор создающих меню с красными буквами на коричневом фоне, невидимые на дисплее с видеоадаптером типа CGA (не говоря уж о монохромном дисплее) или ядовитожелто-красные на синем, видимые даже из другой комнаты;

б). Удобной. Это означает, что в программе должны соблюдаться все сложившиеся стереотипы пользовательского интерфейса: наличие немногих легко запоминаемых "быстрых клавиш", стандартных назначений основных клавиш (Esc, Enter, Home, End), использование для выбора меню, а не цифр — номеров позиций или ответов "Д/Н" или "Y/N". Набор значащих клавиш программы не должен быть большим и "плавающим" (скажем, выход из всех режимов должен происходить всегда при нажатии одной и той же клавиши). Весь диалог с системой в идеале должен состоять из немногих нажатий клавиш управления курсором и клавиш Enter и Esc.

Важность понимания этих элементарных принципов видна, например, из того, что даже многие западные программы, а также многие из современных программ проверки орфографии презирают удобство выбора файла из списка и предлагают раз за разом набирать имя файла вручную, всякий раз злорадно сообщая об ошибке (система WYWTWG — What You Want, That You Get). Об особой любви советских программистов к ответам "Д/Н" при наличии у пользователей клавиатур разных стандартов нечего и говорить.

Обязательно нужно, чтобы программа умела работать с манипулятором "мышь";

в). Программа должна быть понятной. Это достигается благодаря развитой системе пояснений (on-line help) и подробной документации.

В заключение можно добавить, что, по нашему мнению, наилучшая возможная система проверки текстов должна распространяться и на стиль (находить неправильные и тяжелые обороты — излишние местоимения "который", например); проверять согласование подлежащего и сказуемого и правильность управления в предложении одних слов другими (находя ошибки типа "мыть руке"); программа должна предлагать синонимы к любому слову и позволять вставлять их в текст. Это дело будущего.

А пока для выбора программы контроля правописания покупателю нужно подготовить список своих требований к качеству программы проверки текстов, составить набор собственных тестов и заручиться советом профессионального программиста для уверенности в том, что купленная программа будет работать на нужном компьютере. Еще лучше для проверки совместимости программы взять у продавца демонстрационную версию программы и на досуге "поиграть" с ней.

И.Ашманов, Н.Руссова



CLARION — СУБД для профессионалов

СУБД Clarion (фирма Clarion Software) предназначена для высококвалифицированных программистов, создающих сложные программные комплексы. Основными составляющими этой СУБД являются язык программирования для создания приложений и набор утилит для поддержки процесса программирования (основная из них — Designer). Clarion первоначально создавался как средство проектирования интерфейсов. Однако разработка оказалась настолько удачной, что со временем превратилась в интегрированную систему для создания баз данных и средств работы с ними.

По большому счету, Clarion представляет собой язык программирования высокого уровня, предназначенный для создания управленческих, организационных и экономических приложений (часто называемых деловыми). Язык имеет простой и ясный синтаксис. Поддержка экранных форм, отчетов, файлов, запросов к базе данных и т.п. осуществляется встроенными средствами языка, а не отдельными пакетами разработчика, как, например, в dBASE. Clarion является воплощением в современном виде концепции "для каждого класса задач свой язык программирования", и с этой точки зрения его предшественником можно считать КОБОЛ. КОБОЛ, очень хорошо зарекомендовавший себя на больших ЭВМ, хотя и был перенесен на персональные компьютеры, не нашел на них широкого распространения. И это вполне естественно, так как он учитывал специфику разработки не только круга деловых задач, но и техники. При переносе на ПК он был сильно изменен с целью удовлетворения возможностей работы с электронными формами, базами данных, отчетами, сетями ЭВМ. Результатом явился сложный и запутанный синтаксис. За это время изменились также и характер деловых приложений, и требования к программному обеспечению (на главные роли вышли интерактивный режим, активная работа с экраном, широкие возможности графической обработки данных). Кроме того, КОБОЛу, в котором отсутствует модульность, трудно конкурировать с хорошо структурированными языками. Как ни странно, именно коммерциализация производства ПО для ПК послужила одной из причин появления Clarion. На персональных компьютерах работает масса программ, предназначенных для области деловых приложений (электронные таблицы, текстовые процессоры, СУБД, файловые системы, генераторы экспертных систем и т.д.), но в этой сфере максимальный эффект дают программы, созданные по индивидуальному заказу. А коммерческие системы (на создание которых тратятся огромные усилия), обладающие отличным качеством и предназначенные для широкого круга пользователей, не позволяют в каждом отдельном случае достичь максимума желаемого. Большим достоинством этих систем является простота использования и разнообразие возможных приложений. С другой стороны, простота языка неизбежно влечет за со-

бой ограниченность возможностей при попытке написать достаточно сложную прикладную программу.

Бесспорно, фактическим стандартом СУБД персонального компьютера остается dBASE. Основной составляющей этой СУБД является язык управления БД. Этот класс языков появился одновременно с ПК. Акцентируя внимание на работе с дисплеем и дисковой памятью, однозначно сопоставляя записи с экранными формами, эти языки при их применении очень сильно затрудняют разработку действительно сложных программ. Центральную роль в таких СУБД играет Администратор БД, который обычно в интерактивном режиме или с помощью меню выполняет процедурные операторы. Таким образом, работа с БД осуществляется без написания программы. Однако последовательность операторов, записанная в файл, уже становится программой, а операторы Администратора БД становятся языком программирования (хотя сам Администратор БД является интерпретатором). Сейчас dBASE-подобные СУБД вытесняются семейством Paradox. По числу продаж Paradox за первые шесть месяцев 1991 года прочно удерживал первое место. Однако несмотря на многие преимущества перед dBASE, среди которых наиболее впечатляют средства работы с графикой, он остается представителем СУБД того же класса.

Clarion, в отличие от dBASE и Paradox, включает язык программирования в полном смысле этого слова, причем более высокого уровня, чем, скажем, Паскаль или Си. Правда, он ориентирован на определенный, хотя и крайне многочисленный класс задач. Как уже отмечалось, Clarion состоит из собственно языковых средств и ряда утилит. Язык программирования Clarion — это язык высокого уровня, подобный Паскалю, с некоторыми заимствованиями из PL/1, КОБОЛа, Бейсика. В нем имеется набор средств для работы с данными, файлами, управления экраном, построения отчетов. Широко представлены управляющие структуры. Синтаксис языка довольно прост. Написание символов не чувствительно к смене регистра. Первая позиция каждой строки зарезервирована под метку.

Между разработчиками хорошо структурированных языков постоянно идет спор — что лучше, выделение знаками препинания окончания оператора (так сделано в COBOL, ADA, PL/1) или разделение операторов, как в языках Pascal, Modula-2, и какова при этом роль операторных скобок. В Clarion принято очень интересное решение — признаком окончания оператора может служить конец строки, начало комментария или точка с запятой. Оператор может быть перенесен на следующую строчку с помощью символа продолжения. В языке Clarion для завершения группы операторов используется точка. Если группа операторов занимает много строк, она может быть завершена точкой, находящейся непосредственно под последним оператором. Операторы внутри группы должны иметь отступ.

Типы данных представлены тремя видами целых чисел, упакованными вещественными числами переменной длины, вещественными числами с плавающей запятой двойной точности, строками с шаблонами, разнообразными формами представления дат. Все типы данных не нуждаются в специальном преобразовании при их совместном использовании.

В Clarion широко применяется выделение групп операторов, называемых структурами, для резервирования специальных действий. Хотя это явно не оговорено, в Clarion имеются структуры с заранее обусловленными именами, имеющие по умолчанию определенное функциональное назначение. Это, например, SCREEN (работа с экраном), AREA (описание оверлейной области), MODULE (описание модулей-сочленов) и т.п. Структура REPORT предназначена для создания отчетов. Она описывает размер страницы, расположение заголовка, подножия страницы и тела отчета. При форматировании отчетов можно завести структуру отчета, изобразить его компоновку, внести с клавиатуры изменения и в итоге сгенерировать новую структуру отчета в разрабатываемой программе.

Язык Clarion очень компактен — в нем всего 25 зарезервированных слов, которые не могут быть метками. Большинство ключевых слов не являются зарезервированными и могут использоваться в качестве меток, что очень удобно. Так, например, структура описания экрана SCREEN может иметь и метку SCREEN. Но программа PROGRAM не может иметь метку PROGRAM, так как это слово зарезервировано.

Довольно тонким местом при составлении программ на разных языках высокого уровня является разбиение программ на модули. Это связано с необходимостью следить за корректным использованием локальных и глобальных ссылок, а также с организацией связи между модулями. В Clarion программа состоит из основного модуля и модулей-сочленов. Глобальные данные описываются в основном модуле. При наличии в модуле-сочлене глобальных ссылок необходимо в нем указать основной модуль, в котором они были описаны, а не каждую глобальную переменную. Программа, в которой находятся модули-сочлены или процедуры (функции), должна иметь структуру MAP, указывающую на них.

Набор утилит Clarion, входящих в состав СУБД, обеспечивает автоматизацию процесса проектирования и программирования базы данных. В него входят следующие утилиты:

Filer — создание файлов Clarion по тексту исходной программы;

Sorter — управление файлами Clarion;

Scanner — отображение данных на экране в соответствии со структурой БД;

Converter — преобразование файлов из (в) формата Clarion в (из) формат dBASE, DIF, ASCII.

Helper — построение экранов помощи, связанных с полями, меню и другими экранами.

Editor — редактирование текстов программ Clarion, имеются макросредства и обеспечивается прямой доступ к средствам форматирования экранов и отчетов.

Processor — исполнение и отладка программ в режиме интерпретации. Интерактивный отладчик

предоставляет полный набор стандартных возможностей, среди которых такие, как просмотр и изменение переменных, установка контрольных точек, трассировка, повторное выполнение программы с произвольного.

Translator — создание стандартных объектных модулей, соединение их с библиотечными модулями Clarion и Си и создание в итоге EXE-файла. Обычно программы, созданные этой утилитой, требуют меньше памяти и выполняются быстрее, чем созданные утилитой Processor.

Designer — проектирование приложений. Ни одна СУБД не имеет такого мощного средства автоматизации процесса проектирования приложений, как утилита Designer. Эта утилита используется с тремя целями — создание прототипа (макета) прикладной системы, создание полностью готовых приложений и разработка каркаса (ядра) программы, который в дальнейшем будет дополнен более специфическими процедурами (при этом возможна и замена некоторых фрагментов кода, созданного Designer). Целый ряд возможностей (создание экранных форм, форматирование отчетов, обработка файлов и т.п.) Designer и Editor совпадает, но отличие заключается в том, что Designer при этом создает интерфейсное окружение и дает возможность получить полностью законченную программу без программирования. Эта утилита может быть вызвана либо из меню Clarion, либо нажатием зарезервированных функциональных клавиш, либо непосредственно из DOS. Пожалуй, одним из самых больших недостатков Clarion является то, что изменения, внесенные с помощью Editor в текст, созданный посредством Designer, после возврата в Designer видны не будут. Более того, при проведении еще одного сеанса работы с Designer эти изменения полностью исчезнут.

Для создания программы с помощью утилиты Designer необходимо, пользуясь подпунктами меню этой утилиты в интерактивном режиме, описать файлы, экранные формы, формы отчетов, экраны и режимы вывода помощи, а также меню приложения. После этого утилите задается режим создания исходного текста программы на языке Clarion для последующей обработки его компилятором. Откомпилированный текст программы может быть либо представлен в виде выполняемого файла, либо сразу же выполнен run-time-процессором CRUN. Утилита Designer дает эффективный исходный код на языке Clarion.

Необходимо подчеркнуть возможность создания приложений как для стандартной однопользовательской, так и для многопользовательской системы. В Clarion имеются заготовки двух файлов-моделей для этих случаев STANDARD.MDL и NETWORK.MDL соответственно. Designer на основе приведенных описаний объектов создает код, содержащий реализацию как объектов, так и логических связей между ними, и вставляет его в код генерируемой модели.

Взаимодействие утилит Clarion обеспечивает поддержку среды разработки, во многом аналогичную средам общеизвестных языков на ПК. Так, находясь в Editor, нажатием соответствующих клавиш можно обеспечить трансляцию и (или) выполнение редактируемой программы. При обнаружении ошибок трансляции они высвечиваются. Пользователь имеет возможность переходить от ошибки к ошибке и получать достаточно полные объяснения их причин.

Управление экраном в Clarion достаточно гибко. Компоновка экрана описывается структурой SCREEN. В ней в виде строковых констант описаны графические символы и буквы, а в виде строк — экранные переменные. Возможны создание меню, всплывающих окон и организация их взаимодействия с помощью специальных операторов. Большим достоинством является возможность форматирования экрана редактором Clarion, который по заданной компоновке экрана создает экранную структуру, изображает компоновку экрана и, учитывая изменения, внесенные с клавиатуры, заменяет экранную структуру в проектируемой программе.

В Clarion используется общепринятая для СУБД на ПК структура файлов, при которой в файле хранится заголовок, обеспечивающий компоновку файла и записи, а также сами записи. Ключи записей находятся в отдельном файле. Данные в файлах Clarion хранятся с высокой надежностью, так как при отсутствии к нему запроса файл является логически закрытым. Кроме того, средства Clarion дают возможность предусмотреть в программе автоматическое восстановление файла при повторном выполнении программы, если ему предшествовал сбой в системе. Возможна защита обработки запросов.

Последовательность обработки исходных файлов Clarion (расширение CLA) следующая. Транслятор переводит исходный код в псевдокод, соответствующий командам виртуальной машины Clarion. Файлы с псевдокодом имеют расширение PRO, указывающее, что они предназначены для обработки утилитой Processor, которая сформирует обычный объектный файл с расширением OBJ. Одновременно создаются файл с адресами символов (тип SYM), с листингом трансляции (тип LST) и файл ошибок трансляции (тип ERR). Процесс трансляции всей программы организуется так, что при трансляции модуля-сочлена загружается файл символов и сразу транслируются адреса глобальных данных. Это приводит к тому, что отпадает необходимость в компоновке. Однако изменение глобальных данных в программном модуле требует повторной трансляции всех используемых модулей-сочленов. Естественно, что программный модуль и модули-сочлены транслируются отдельно. Программный модуль начинается оператором PROGRAM и является головным файлом, в котором находятся описания глобальных переменных, процедуры, функции, структура MAP и сам текст программы.

Структура MAP обязательна, если в программном модуле есть процедуры, функции или модули-сочлены, так как она обеспечивает компоновку всей программы. В модулях-сочленах MAP присутствовать не может.

Структура MAP может состоять из операторов PROC, FUNC, MODULE, AREA и OVERLAY. Структура MODULE предназначается для объявления модулей-сочленов со всеми их внутренними процедурами и функциями. Эта структура обязательно должна быть внутренней по отношению к структуре MAP. Внутри программы, процедуры или функции оператор CODE отделяет раздел данных, в котором содержатся операторы описания, от раздела исполняемых операторов. При организации взаимодействия процедур и функций локальные параметры передаются по значению, а внешние параметры — по адресу. Функции в Clarion делятся на два типа: библиотечные функции, встроенные в язык Clarion, и функции, написанные пользователем, которые должны быть указаны как параметр оператора FUNC в структуре MAP. Аналогично процедуры должны быть указаны в качестве параметра оператора PROC в структуре MAP. Первым оператором программного модуля всегда служит оператор PROGRAM. Модуль-сочлен является файлом с операторами, первый из которых — MEMBER. Директивы транслятору могут находиться в произвольном месте программы. Операторы описания могут определять не только константы и переменные, но и файлы, отчеты, экраны, таблицы памяти. Локальные подпрограммы не содержат операторов описания, не имеют параметров, и обращение к ним возможно только из содержащей их программы, процедуры или функции. Язык содержит многочисленные функции для работы с клавиатурой, дисплеем, файлами. Возможно расширение языка программами, написанными на других языках программирования.

Операторы обработки экранов взаимодействуют непосредственно с видеопамятью ПК. Структура экрана в Clarion определяется экранной формой, которая состоит из изображаемых символов, видеоатрибутов и полей ввода информации. Экранная форма описывается структурой SCREEN. Она начинается с одноименного оператора. Эта структура активизируется оператором OPEN и завершается — CLOSE. Концепция работы с экраном в Clarion сочетает удобство программирования с эффективным использованием возможностей монитора. Первое обеспечивается возможностью создавать экранную форму в редакторе Clarion. Предусмотрена возможность непосредственной организации окон и меню. Для работы с отчетами в Clarion служит структура REPORT, которая содержит форматы и атрибуты, определяющие оформление страниц. Как и при работе с экраном, структура REPORT может создаваться и форматироваться в редакторе.

Хорошо известно, что далеко не каждая программа занимает всю память, отведенную под нее. В Clarion есть средства для экономии такой избыточной памяти. В Clarion имеются возможности доступа к файлам DOS. Некоторые команды Clarion и DOS для работы с файлами совпадают. Для ряда команд DOS в Clarion имеются аналоги. В Clarion при организации работы с памятью предусмотрен такой необычный объект, как таблицы памяти. Они предназначены для хранения элементов, между которыми задан порядок следования, в памяти, оставшейся свободной после загрузки программы. Элементы по отношению к таблицам памяти играют такую же роль, как записи по отношению к файлу. С точки зрения структур данных таблицы памяти представляют двунаправленный список, т.е. список, в котором каждый элемент списка имеет ссылку на предыдущий и на последующий элементы. Элементы такого списка не обязательно должны располагаться в памяти подряд, что предоставляет широкие возможности по работе с памятью. По существу, это концепция динамической памяти с возможностями получать память из "кучи" и возвращать ее обратно. Таблица памяти может содержать до 65536 элементов длиной до 65520 байт каждый. Таблицы памяти описываются структурой TABLE, а доступ к ним обеспечивается при помощи указателя элемента.

В Clarion имеется богатый набор математических функций с достаточно простым синтаксисом.

В СУБД Clarion допускается обработка транзакций в сети. Асинхронные взаимодействия, обработка файлов формата dBASE III обеспечиваются программой Language Extension Modules.

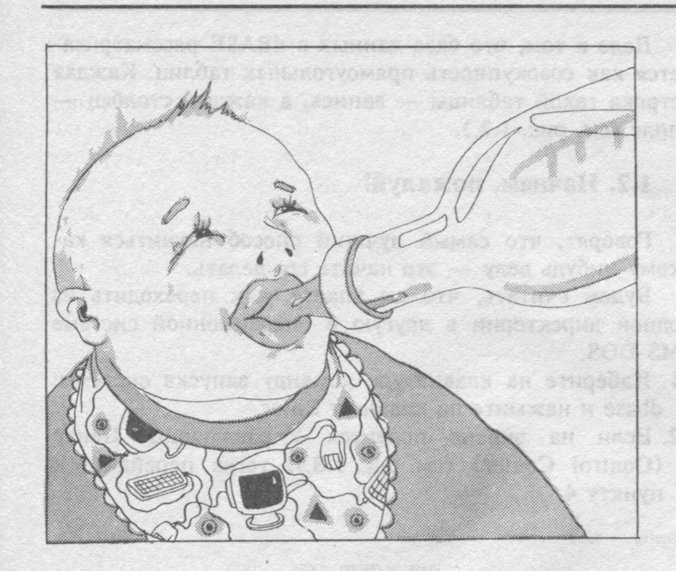
Следуя Кернигану и Риччи, которые считают, что первой программой при встрече с любым новым языком должна быть та, которая выдает на экран некоторые слова типа "Hello, word", приведем ее.

```
TITLE ('FIRST PROGRAMM')
HELLO PROGRAM
LAYOUT SCREEN
    ROW(15,40)STRING('Hello,')
    ROW(18,45)STRING('word!')
CODE
OPEN(LAYOUT)
RETURN
```

В этой программе первая строчка задает заголовок листинга трансляции, а вторая содержит метку (имя) программы и оператор начала программного модуля. Структура SCREEN описывает экранную форму и имеет метку LAYOUT. Строка Hello начинается на экране с сорокового столбца пятнадцатой строки, а строка word — с 45 столбца восемнадцатой строки. На этом раздел описаний заканчивается. Раздел кодов состоит из операторов OPEN, который выводит созданную экранную форму на экран, и RETURN, который завершает выполнение программы.

Приложения, созданные в Clarion, имеют высокую эффективность и надежность. Сообщения об ошибках исчерпывающи и понятны, а встроенная консультационная система обеспечивает достаточный уровень для сопровождения пользователя в процессе работы.

к. ф-м. н. А.Смородинский



Никогда не поздно начать с начала. И если вы только что купили персональный компьютер, а к нему хорошее программное обеспечение (у легальных поставщиков), то не плохо было бы начать обучение прямо сейчас. И поможет вам не кто иной, как КомпьютерПресс. С этого выпуска мы начинаем публикацию серии уроков по dBASE IV для начинающих. Успехов вам!

dBASE IV для начинающих

Серия статей предназначена в первую очередь для обучения пользователей работе с системой. Мы не будем касаться вопросов написания программ, так как это дело профессионалов и лучше программиста программу все равно не напишешь. Другой разговор, что при наличии DBASE IV нам и не нужны программисты. Тем более, что прежде, чем написать даже небольшую программу, они будут долго ходить к вам и спрашивать как бы вы хотели, чтобы эта программа работала. Вы будете долго объяснять им, чего бы желали получить. В результате вы разойдетесь в полной уверенности, что все будет хорошо, но когда через несколько месяцев программисты придут и покажут, что же все-таки получилось, программа будет работать совсем не так, как вы себе это представляли. Вы будете недовольны, но вам докажут, что все сделанное полностью соответствует техническому заданию, которое вы подписали после обсуждения (а вы просто были очень заняты и некогда было вникать в смысл этой бумаги, тем более, что от обилия специальных терминов у вас закружилась голова). Вам скажут, что доработки потребуют времени и денег. И в конце концов вам придется работать с несимпатичным средством или же просто не пользоваться им.

Сколько потрачено нервов и денег! Ну, деньги-то ладно, а нервные клетки не восстанавливаются.

Конечно, если потребуется сложная программа, все равно придется обращаться к программистам. Но могу открыть маленький секрет: сложную программу хорошо написать могут только настоящие профессионалы, а с ними и хлопот будет меньше — опыт у них большой.

Но для достаточно простой системы (например, личная адресная книга, библиографический каталог или телефонный справочник) программа и не потребуется. Изучив работу с системой DBASE IV, вы все сможете сделать сами. Тем более, что для овладения основами работы с этим пакетом программ не потребуется много времени. В большинстве глав разбирается один и тот же пример, поэтому по окончании обучения вы будете иметь на своем компьютере персональную справочную систему.

Итак, если вы решились, вперед!

Глава 1. Первоначальные сведения о базах данных

1.1. Что такое база данных и система управления ею

База данных — это особым образом организованная совокупность взаимосвязанных, хранящихся вместе данных. Примерами баз данных могут служить адресная книга, картотека, поваренная книга, словарь и энциклопедия.

Давайте поближе рассмотрим такую простейшую базу данных, как ваша телефонная записная книжка,

которой вы пользуетесь каждый день. В нее заносятся фамилии, номера телефонов и адреса абонентов, а также некоторые дополнительные сведения.

Для этих данных в записной книжке отводится отдельная строка (см. рис.1.1). Пользуясь терминологией баз данных, каждая строка телефонной книжки, соответствующая информации об одном человеке, называется записью. Каждая запись состоит из полей (фамилия, телефон и т. д.). Совокупность всех записей составляет базу данных или файл базы данных (в дальнейшем файл). Список полей — это структура базы данных.

Фамилия	Адрес	Телефон
Кузнецов И.И.	Кутузовский пр., д. 17, кв. 6	145-23-18
Лунин А.М.	пр. Мира, д. 12, кв. 23	356-13-12
...		

Рис. 1.1. База данных (телефонная книжка)

Ваша телефонная книжка устроена определенным образом — она позволяет заносить сведения о ваших знакомых (на бумагу с помощью ручки), быстро отыскивать нужную вам фамилию с помощью алфавита. Можно сказать, что ручка и алфавит управляют вашей базой данных (телефонной книжкой).

Таким образом, вы ежедневно много раз пользовались базами данных. Вы искали в них информацию, добавляли, изменяли или удаляли ее. Другими словами, вы управляли этой информацией. И можно сказать, что вы опытный пользователь.

Система управления базами данных (СУБД) — это средство управления данными: она обеспечивает возможность изменения и добавления данных и поиск конкретных элементов. DBASE IV — это система управления реляционной базой данных. Этот термин, "реляционная", требует пояснений.

поле 1 поле 2 поле 3 ... поле n	
...	запись 1
...	запись 2
...	....
...	запись m

Рис. 1.2. Таблица реляционной базы данных

В терминах СУБД подобные таблицы называются отношениями (отношение — relation, англ.). Отсюда и название модели базы данных — реляционная.

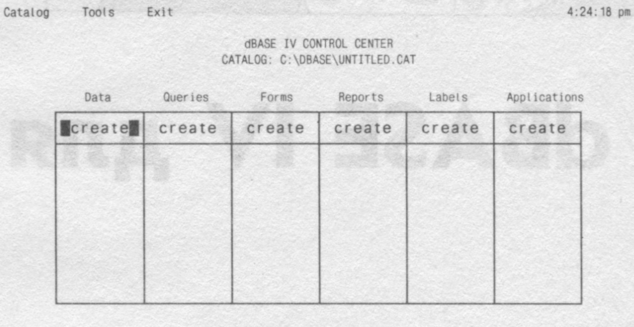
Дело в том, что база данных в dBASE рассматривается как совокупность прямоугольных таблиц. Каждая строка такой таблицы — запись, а каждый столбец — поле (см. рис. 1.2.).

1.2. Начнем, пожалуй!

Говорят, что самый лучший способ научиться какому-нибудь делу — это начать его делать.

Будем считать, что вы знаете, как переходить из одной директории в другую в операционной системе MS-DOS.

1. Наберите на клавиатуре команду запуска системы: dbase и нажмите на клавишу Enter.
2. Если на экране появится Управляющий Центр (Control Center) (см. рис. 1.3.), тогда перейдите к пункту 4.


Catalog Tools Exit 4:24:18 pm

dBASE IV CONTROL CENTER
CATALOG: C:\DBASE\UNTITLED.CAT

Data	Queries	Forms	Reports	Labels	Applications
create	create	create	create	create	create

File: New File
Description: Press ENTER on <create> to create a new file

Help:F1 Use: Data:F2 Design:Shift-F2 Quick Report:Shift-F9 Menus: F10

Рис. 1.3. Управляющий центр dBASE

3. Если ваша система установлена так, что при загрузке выводится приглашение для набора команд dBASE (командная строка — точка над строкой состояния, см. рис. 1.4), то нужно набрать на клавиатуре команду assist и нажать клавишу Enter.

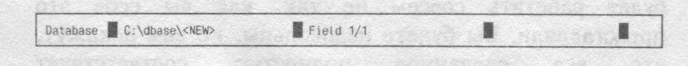

Database C:\dbase\<NEW> Field 1/1

Рис. 1.4. Командная строка dBASE

4. Теперь рассмотрим, что из себя представляет управляющий центр.

1.3. Приемы работы с управляющим центром

Перед вами на экране шесть панелей. В панелях содержатся списки файлов различных типов. Таких типов — шесть. Столько же, сколько и панелей:
1) панель Data — файлы базы данных (*.dbf);

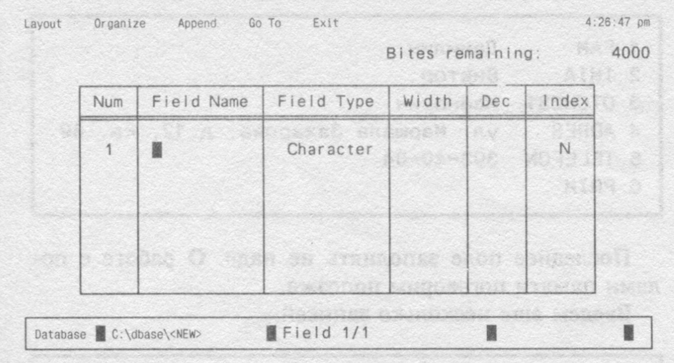


Рис. 1.5. Экран проектирования баз данных

2) панель Queries — файлы запросов (расширение .que). Содержат инструкции для манипулирования данными;
3) панель Forms — файлы форм (расширение .frm). Используются для ввода, редактирования и просмотра информации;
4) панель Reports — файлы отчетов (.fro). Эти файлы содержат отчеты по данным базы. Данные выводятся на экран или печатающее устройство в специальном формате;
5) панель Labels — файлы этикеток (.lbl). Файлы этого типа очень похожи на файлы предыдущей панели, за исключением того, что отчет выводится на печать в виде этикеток (карточек);
6) панель Application — приложения. Эти файлы содержат написанные программы, которые выполняются СУБД.

Во всех панелях вы можете сделать два выбора: создать новый файл, выбрав «create», или открыть уже созданный файл.

Панель Data (данные) является в данный момент активной, на маркере «create» стоит курсор (курсор — подсвеченный прямоугольник). Для того чтобы перейти в другую панель, вы можете нажать на клавишу «стрелка вправо», и курсор передвинется на панель запросов. Нажимая таким образом на эту клавишу несколько раз, мы можем пройти все панели управляющего центра и вернуться обратно. Курсор фактически движется по кругу, его движение зациклено. Для того чтобы перейти, например, из панели Данных в панель Приложений, не обязательно пять раз нажимать клавишу «стрелка вправо», можно перейти в нужную панель, нажав только один раз клавишу «стрелка влево».

Но вернемся к панели данных. Теперь мы попробуем создать файл базы данных для нашей телефонной записной книжки.

Num	Field Name	Field Type	Width	Dec	Index
1	FAM	Character	20		N
2	IMIA	Character	12		N
3	OTCHEST	Character	15		N
4	ADRES	Character	50		N
5	TELEFON	Character	7		N
6	PRIM	Memo	10		N

Рис. 1.6. Информация о полях базы данных

1.4. Создание файла базы данных телефонной книжки

1. Установив курсор на маркер «create» панели Data, нажмите клавишу «Enter». Перед вами появится экран проектирования базы данных (рис. 1.5.) с курсором в колонке Field Name (имя поля).
2. Наберите на клавиатуре название первого поля нашей телефонной книжки (латинскими буквами): FAM (фамилия).
3. Нажмите «Enter» для перехода к следующей колонке Field Type (тип поля).

В dBASE поля могут быть шести типов:
- символьное (Character) — любые символы, величина поля — 274 символа.
- числовое (Number) — любые числа, величина поля до 20 знаков;
- числовое поле с плавающей точкой (Float). Используется в основном в научных приложениях;
- логическое (Logical) — длина поля — 1 символ. Содержит или знак T (true — истина), или F (false-ложь);
- поле даты (Date) имеет длину 8 символов, содержит день, месяц и год, причем в специальном формате, например, 21 августа 1991 года будет представлено в dBASE как 21.08.91. Точки называются разделителями (в качестве разделителя можно использовать и знак "/", в зависимости от того, как установлено в dBASE).
- поле памяти (memo). Это текстовое поле может быть использовано для написания больших документов, ограниченных только размерами свободной памяти. Данные будут размещены в дополнительном файле с расширением DBT.

По умолчанию в dBASE IV тип поля является символьным (Character), поэтому это значение уже появилось в поле ввода.

4. Нажмите клавишу «Enter». Курсор перейдет в колонку Width (ширина поля).
5. Наберите 20 и нажмите «Enter». Курсор автоматически перейдет в колонку Index (об индексе попозже), перескочив через колонку Dec (десятичное), которая заполняется только для числовых полей.
6. Не беспокойтесь сейчас об индексе и нажмите «enter». В колонке Num (номер) появляется число 2, и курсор передвинется на следующую линию. Вы можете вводить информацию о втором поле.

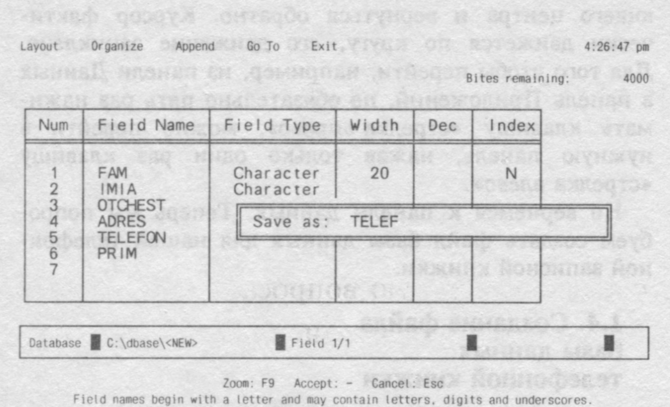


Рис. 1.7. Сохранение структуры файла .dbf на диске

7. Повторите шаги 1-6 для ввода информации о полях (см. рис. 1.6.).

8. Нажмите клавишу «Enter», когда курсор окажется в седьмом пустом поле. Перед вами возникнет приглашение для ввода названия файла (Save as — Сохранить как) (рис. 1.7.). Введите имя вашего файла — TELEF и нажмите «Enter».

9. Внизу экрана появится вопрос dBASE к вам:

Input data records now? (Y/N)
Вводить данные сейчас? (Да/Нет)

10. Нажмите Y (Да).

1.5. Ввод данных

Перед нами экран (см. рис. 1.8.). Он называется экраном режима Редактирования (Edit).

Появится пустая запись с курсором в поле FAM, готовая для ввода информации. Введем первую запись (после ввода информации в каждое поле не забудьте нажимать Enter):

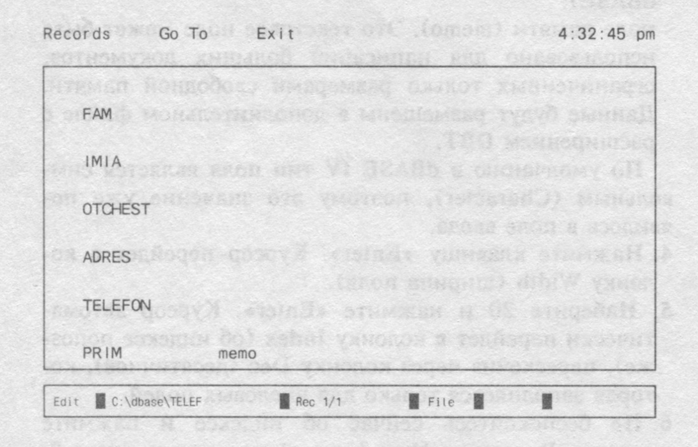


Рис. 1.8. Экран режима редактирования dBASE

1	FAM	Данилин
2	IMIA	Виктор
3	OTCHEST	Иванович
4	ADRES	ул. Маршала Захарова, д.12, кв. 89
5	TELEFON	393-40-04
6	PRIM	

Последнее поле заполнять не надо. О работе с полями памяти поговорим попозже.

Введем еще несколько записей:

1	FAM	Большакова
2	IMIA	Татьяна
3	OTCHEST	Александровна
4	ADRES	ул. 13 Парковая, д.22, кв. 3
5	TELEFON	464-35-43
6	PRIM	

1	FAM	Ушакова
2	IMIA	Елена
3	OTCHEST	Алексеевна
4	ADRES	Армянский пер., д.10, кв. 3
5	TELEFON	385-15-12
6	PRIM	

Маленькое замечание, прежде чем вы будете вводить следующую запись. В dBASE IV русские заглавные буквы "К" и "Н" не вводятся, вместо них надо использовать их латинские эквиваленты — "К" и "Н". Это маленькое неудобство, но с ним приходится мириться.

1	FAM	Кожухова
2	IMIA	Ольга
3	OTCHEST	Альбертовна
4	ADRES	ул. Сталеваров, д.10, кв. 15
5	TELEFON	404-35-04
6	PRIM	

1	FAM	Чесаков
2	IMIA	Давид
3	OTCHEST	Моисеевич
4	ADRES	ул. Тверская, д.10, кв. 50
5	TELEFON	285-56-82
6	PRIM	

После того как вы ввели последнюю запись, нажмите на комбинацию клавиш Ctrl-End для сохранения информации в вашем файле данных.

А.Иванов

(продолжение следует)

КОНКУРС
«ЛУЧШАЯ ПУБЛИКАЦИЯ 1992 ГОДА»

Агентство «КомпьютерПресс» в 1992 году продолжает конкурс на лучшую публикацию, посвященную вопросам применения вычислительной техники в нашей стране и за рубежом.

Победителей ждут премии:

1 премия — 5000 рублей

2 премия — 3000 рублей

3 премия — 1000 рублей

10 поощрительных премий — годовая подписка на журнал «КомпьютерПресс».

На конкурс принимаются статьи объемом до 2 авторских листов (80 Кбайт) в машинописном виде или на дискете в формате Microsoft Word или ASCII. В конце статьи необходимо указать список использованных источников в виде: автор, название источников на языке оигинала, месяц и год издания.

Лучшие работы будут опубликованы на страницах «КомпьютерПресс».

Работы присылать по адресу: 113093 Москва, а/я 37.

Переводы на конкурс не принимаются.

В конверт следует вложить лист с указанием фамилии и имени автора, обратного адреса, телефона, ученой степени, краткого описания (не более 100 символов) сферы научных и технических интересов.
Желательно приложить копии использованных материалов.



Я отнюдь не думал предлагать общему вниманию компилятивные биографии и еще менее хотел выдавать свои фантазии за исторические исследования.

Мальчик-миллиардер из Microsoft

Часто ли у вас бывают отпуска? Всего раз в год? Но зато сколько удовольствий и воспоминаний!

Для 36-летнего председателя совета директоров Microsoft отпуск — всегда проблема. Когда-то он считал невозможной даже мысль об отдыхе, сейчас он, хоть и неохотно, порой отправляется в увеселительные поездки, но только предварительно определив их цель. "В одном из наших путешествий мы большую часть времени занимались физикой, — вспоминает в интервью журналу Playboy бывшая подруга Гейтса Энн Уинблад, — Мы слушали кассеты с записями лекций Ричарда Фейнманна и читали всевозможные книги по этому предмету". А в солнечной Бразилии Билл Гейтс посвятил отпуск изучению "Молекулярной биологии гена" Джеймса Уотсона.

Возможно, целеустремленность, которая заставляет Гейтса проводить отпуск будто в читальном зале университета, и сделала его и возглавляемую им компанию самой мощной силой в мире программного обеспечения. В 1991 году объемы продаж Microsoft должны были достичь 1,6 миллиарда долларов — больше, чем у четырех ее главных конкурентов, вместе взятых. "Он — наиболее влиятельный человек в компьютерной индустрии", — писала газета Wall Street Journal. "Гейтс напоминает мне промышленных магнатов XIX века, силой своей воли и деловым талантом построивших нефтяные, стальные и банковские корпорации", — отмечал обозреватель Стюарт Элсоп.

Но в компьютерной индустрии масса людей считает, что Гейтс даже чересчур удачлив, и, конечно, его достижения не могли не дать обильной пищи для злых языков. "Билл Гейтс страдает манией величия, — заявил в интервью газете Los Angeles Times один из производителей программного обеспечения. — Он хочет быть первым во всем, за что берется". "Билл хочет оттяпать такой большой кусок программной индустрии, какой только может проглотить. А у него очень хороший аппетит", — заметил другой. "Иногда Microsoft наваливается всей своей тяжестью, не очень-то связывая себя соображениями морали. Но просто быть преуспевающим еще не противоречит закону", — добавил третий.

Наслушавшись подобных возмущенных возгласов противников, легко прийти к мнению, что Гейтс — это барон-грабитель века информатики. Но не все соглашаются с подобными оценками.

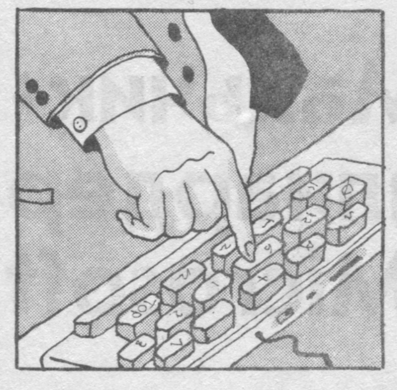
Феличе Бальзамо старалась взглянуть на маленькое существо, лежавшее около нее на широкой купеческой кровати, и говорила мужу:

— Смотри, Пьетро, какие блистающие глаза у малютки, какой ум написан у него на лобике!.. Наверное, он будет если не кардиналом, то, во всяком случае, полковником!..

Синьор Бальзамо, на минуту оторвавшись от большой расходной книги и засунув за ухо перо, повернулся к кровати и, не подходя к ней, ответил:

— Вероятнее всего он будет честным купцом, как его отец и дед. Может быть, впрочем, он будет адвокатом; это теперь выгодное занятие.

“Он на треть Эйнштейн, на треть Джон Макинрой и на треть генерал Паттон (командующий войсками союзников в Северной Африке, славившийся своей экстравагантностью)“, — говорит Хейди Ройзен, один из конкурентов Гейтса. Когда корреспондент Playboy Дэвид Рензин сообщил об этой характеристике Гейтсу, тот улыбнулся: “Это большой комплимент. Должно быть, этот человек ко мне хорошо относится“.



Билл Гейтс родился в состоятельной семье. Его отец Уильям имеет в Сиэтле юридическую практику, а мать Мэри возглавляет банк. Злые языки утверждают, что главой семейства является мать, “самая умная из Гейтсов“, которой Билл во многом обязан своими нынешними успехами.

Билл закончил престижную частную школу, где в математике опережал даже старших. Там он познакомился с Полом Алленом, сойдясь с ним на почве увлечения фантастикой. Позже в местном вычислительном центре был создан детский кружок, в который записались Пол и Билл, после чего они начали безбожно прогуливать уроки, проводя все время (даже ночью) у вычислительных машин. В восьмом классе Билл “со товарищи“ уже зарабатывали деньги, создавая программы, требующиеся школе. Вырученные деньги шли на закупку дополнительного машинного времени в том же центре.

Свой выпускной класс Гейтс практически целиком прогулял: в это время он уже работал программистом в аэрокосмической корпорации TRW, получая 20 тысяч долларов в год. Вспоминая сегодня о том времени, он говорит: “В фирме TRW я встретил человека по имени Нортон. Он всегда мне указывал на то, что я делал не очень хорошо. Так что, если я ленюсь или делаю что-то недостаточно тщательно, я представляю, что он сейчас подойдет, посмотрит и скажет: “Смотри, это можно сделать несколько лучше“.

В 1973 году Билл поступил в Гарвардский университет, туда же из Вашингтонского университета перевелся Пол. Там они и прочитали в журнале Popular Electronics статью о настольном компьютере Altair, основанном на разработанном фирмой Intel процессоре 8080. Объем памяти составлял 4 Кбайта, и производитель, компания MITS, нуждалась в компьютерном языке, чтобы сделать машину программируемой.

Кто-то другой, вероятно, подумал бы, но Пол и Билл тут же позвонили в MITS и сказали, что у них есть очень компактная версия Бейсика. К счастью для них, MITS назначила встречу для испытания “их“ Бейсика через три недели. Теперь оставалось всего ничего: написать эту версию. Через три недели они летели на встречу, в самолете дописывая свою программу. Допусти они единственную ошибку, возможно, сегодня никто не слышал бы о фирме Microsoft, о Word, о Windows и MS-DOS, но программа сработала, ошибок не было!

Мальчик рос, как растут все дети небогатых купцов, хотя родители ему предоставляли больше свободы, чем это принято. И характер имел смелый, предприимчивый и открытый, не без некоторой вспыльчивости, впрочем. Был скор на руку и крайне самолюбив.

В 1975 году Гейтс бросил учебу в Гарварде. Вместе с Алленом они переехали в штат Нью-Мексико, где основали Microsoft. Первоначально Аллен настойчиво предлагал специализироваться на аппаратном обеспечении, но Гейтс его переубедил, считая, что двигателем компьютерной индустрии будет программное обеспечение. Билл не скрывает, что тогда он принял решение из-за очень простой вещи: “Для создания программы нужны только мозги, а для создания машины еще и какие-то железки“. Вскоре фирма MITS прогорела, а Microsoft получила от IBM предложение о сотрудничестве, и началась Эра Персональных Компьютеров.

В 1980 году IBM обратилась к Microsoft с предложением о создании базовой операционной системы для всех персональных компьютеров, выпускаемых IBM. В то время в Microsoft уже работало 38 человек, которые занимались разработкой и поставкой IBM языков программирования, но операционной системы у них не было и в ближайшее время появиться не могло. Гейтс был вынужден, скрепя сердце, отослать IBM к своему конкуренту — фирме Digital Research, уже тогда хорошо зарекомендовавшей себя своей операционной системой CP/M, действующей на многих 8-разрядных компьютерах. Но вместе с рекомендациями обратиться к Digital Research Гейтс направил IBM и пространный труд, в котором убеждал IBM использовать в своем новом компьютере более мощный 16-разрядный микропроцессор 8088, разработанный к тому времени фирмой Intel. И это сработало! В результате Digital Research была вынуждена, не сумев заключить договор на поставку старой, начать разработку новой операционной системы, а Гейтс получил шанс воспользоваться паузой. И он им воспользовался! В первый раз повезло, когда Пол Аллен нашел операционную систему небольшой фирмы Seattle Computer Products, которая понятия не имела, что эта система позарез нужна IBM. Второй раз повезло, когда через шесть месяцев IBM выразила готовность подписать контракт на поставку новой операционной системы, и в этот момент президента Digital Research не оказалось в Штатах — он был в отпуске где-то в Европе; опять же добрые

люди говорят, что в этот отпуск он уехал не без участия Гейтса... Но, как бы там ни было, между этими двумя "везениями" Гейтс объявил своим близким, что в ближайшие шесть месяцев они его не увидят, поскольку он вынужден будет работать 24 часа в сутки над доводкой новой операционной системы. В результате контракт с IBM был заключен!

В 1981 году IBM сделала MS-DOS сердцем новой машины. В соответствии с договором Microsoft стала получать процент с каждого проданного персонального компьютера.

Калиостро минуло тридцать три года. Он придавал большое значение этому обстоятельству, думая, что это „ время его выступления на историческую арену, и считая жизнь до этого года лишь за подготовку, да и то, может быть, недостаточную к этому шагу.

Сегодня MS-DOS управляет более чем 60 миллионами машин во всем мире. Получаемые Microsoft проценты оцениваются в 200 миллионов долларов ежегодно.

Но Гейтс не хотел зависеть только от IBM. Собравшись с духом, он написал очередной труд и убедил IBM в целесообразности продаж MS-DOS по лицензии другим производителям персональных компьютеров, аргументируя это тем, что ее распространение поможет бороться с влиянием главного конкурента — Apple Computer. Но и этого ему показалось мало. Он убедил некоторых из ведущих производителей персоналок, в частности Compaq, делать их продукцию ПО-НАСТОЯЩЕМУ совместимой с продукцией IBM, то есть, чтобы любая программа, написанная для IBM, могла быть использована на этих компьютерах. В результате этих "убеждений" MS-DOS стали скупать в фантастических количествах. Изрядный кусок рынка был проглочен.

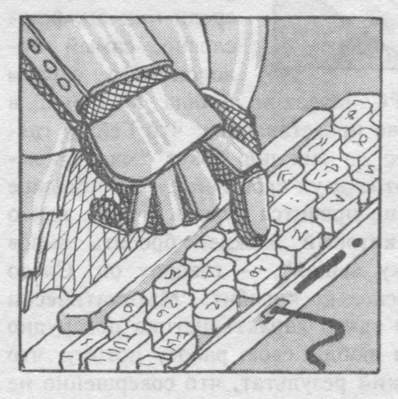
Могущество Гейтса в области программного обеспечения почти беспредельно, но, по его словам, Microsoft не представляет никакой угрозы для конкурентов. Впрочем, судите сами. После появления на рынке Windows 3.0, делающей компьютер вполне доступным для непрофессионала, Apple подала в суд на Microsoft, обвиняя ее в нарушении авторских прав. Да и IBM по этому поводу отнюдь не в восторге. До недавнего времени Microsoft помогала IBM в работе над операционной системой OS/2, которая, как и Windows 3.0, более наглядна для пользователя. Но после появления Windows 3.0, когда за год было продано 3 миллиона экземпляров, тогда как с момента завершения разработки в 1987 году удалось продать только 300 тысяч пакетов OS/2, этому сотрудничеству пришел конец. Сам Гейтс, отметая все обвинения, как не имеющие оснований, скромно говорит, что все дело в том, что "Microsoft сильно изменила мир".

Журнал Business Month писал, что своим успехом Microsoft обязана главным образом удаче и способности Гейтса "нащупывать слабые места более неповоротливых соперников". Этот журнал приводит также слова неназванного "крупного деятеля компьютерной индустрии" о том, что Гейтс — не новатор. Он отыскивает чужие идеи и доводит их до совершенства. Впрочем, подобные обвинения раздавались и по адресу японских компаний, теснящих конкурентов по всей линии фронта.

Сотрудники же фирмы Гейтса придерживаются несколько иной точки зрения. "Он просто гораздо умнее, чем кто-либо другой. Мы имеем дело не с обычным смертным, а с гением. Он способен перерабатывать огромный объем информации и предметно вести беседу практически на любую тему", — говорит Пол Моритц, один из ведущих программистов компании.

Командный дух в Microsoft действительно развит сильно. Хотя сотрудники фирмы зарабатывают меньше, чем могли бы в других местах, и у них не лимитирован рабочий день, текучести кадров в компании почти нет. Причин несколько — от возможности льготного приобретения акций до стремления быть в команде, поставившей перед собой цель достичь доминирующего положения в мире в сфере программирования и избравшей своим девизом "Написать программу, которая бы заставила поставить компьютер в каждом доме".

Со своими сотрудниками Гейтс особо не церемонится. Если кто-то приносит ему на просмотр сырую программу, он может ответить по-простому: "Ты что, дурак?". Но сотрудники не обижаются, поскольку подобный вопрос без серьезных оснований шеф не задает. "Он часто мочалит нас, но главное — стерпеть и правильно ответить. Если ты стушуешься, он перестанет тебя уважать. Таковы правила игры", — говорит программист Джефф Харберс.



Хотя Гейтс несет массу обязанностей в качестве главного администратора, ему ничто не приносит такого наслаждения, как разбор с программистами какой-то проблемы. Как заметил в интервью журналу Playboy Моритц, "его уважают не потому, что он Билл Гейтс, а потому, что он может дать каждому из ребят сто очков в профессиональных вопросах".

Сам Гейтс так формулирует качества настоящего программиста: "Настоящий программист никогда не думает: "Я заработаю кучу денег" или "Я продам сто тысяч копий". Просто такие мысли ничем не помогают работе над программой. Настоящий программист думает: "Нужно ли переписать всю подпрограмму, чтобы вместо трех человек ее могли прочитать четверо? Можно ли сделать ее на 10 процентов быстрее?

Нужно ли еще раз обдумать, как вызвать эту проверку?" Если вы великий программист, то вы делаете все программы взаимосвязанными, поэтому небольшая ошибка может испортить всю работу. Именно поэтому нужно иметь очень стройное мышление, а также желание вернуться назад и что-то изменить. Когда я думаю над структурой программы, и что-то получается, я чувствую себя просто здорово. Я люблю подождать и подготовить хорошую базу до того, как насладиться кодированием и видеть, что все нормально. Это вроде того, когда оставляешь самый вкусный кусок на тарелке напоследок. Иногда я завидую своим коллегам, которым нужно заниматься только своей программой".

Чья карета, красная, с пестро намалеванным гербом, изображавшим в одном углу лазурного поля золотую куропатку, мчалась по Страсбургу с шести часов утра до поздней иногда ночи? На чьем пути нищие останавливались, крестились и благословляли небо? Чей салон самый оживленный, самый многолюдный, где бывают графы, кардиналы, базельские банкиры и богатые еврейки? Чье имя служит городской гордостью?



Если Билл Гейтс завидует только своим программистам, то, видимо, ему завидуют очень многие. Ведь на сегодняшний день Гейтс — девятый среди самых богатых людей Америки. К тому же среди них он, безусловно, самый молодой. По словам издателя журнала PC Magazine Билла Макрона, в Америке очень живуче представление, что Гейтсу где-то около 19 лет. Его до сих пор зовут "мальчиком-миллиардером". Из этих двух слов Гейтсу явно больше нравится первое. "Мне нравится думать о себе, как о молодом человеке, который жаждет бросить вызов устоявшемуся порядку вещей", — говорит он. Слово "миллиардер" его несколько коробит. "Математически это... гм... верно. Но такая характеристика подспудно подразумевает, что я люблю свою работу потому, что она дает экономический результат, что совершенно не так. Радость, подлинный кайф приносит сознание того, что ты смог создать компанию, и это принесло очень хорошие плоды".

Однако внешне в жизни Гейтса эти плоды абсолютно не видны. Он живет по-спартански в скромном домике в Сиэтле. Он почти не смотрит телевизор, хотя и делает исключение для фильмов, выигравших премию Оскар, и видеокассет с записями университетских лекций по тому или иному предмету. По вечерам, после 12-15-часового рабочего дня он читает. Мальчик-миллиардер может наизусть цитировать целые страницы из "Над пропастью во ржи" или "Великого Гэтсби". Если автор его увлекает, он прочитывает его целиком. Еда большей частью поглощается им за рабочим столом. Чтобы поужинать дома, он берет чаще всего навынос пиццу или спагетти. С годами, как положено миллиардеру, он выработал в себе одну утонченную привычку — любовь к французскому шампанскому DomPetrignon, в его холодильнике всегда хранится полдюжины бутылок этого сорта.

Поскольку мальчик до сих пор не женат, дома, как и в офисе, страшнейший беспорядок: неубранная постель — привычка еще со студенческих лет, разбросанные кассеты с записями Chicago, золотой кубок за победу в национальной олимпиаде в области науки и техники в 1990 году и много-много каких-то безделушек. На стене в офисе групповой портрет той восьмерки, которая стояла у истоков Microsoft в семидесятые годы. Выглядят они как хиппи, Гейтс, которому не дашь больше тринадцати, сидит в левом нижнем углу.

Его конкуренты ждут-не-дождутся, когда он женится. Но Гейтс, как он сам говорит, не торопится с этим, и, ухмыляясь, добавляет, что с кем потанцевать ему всегда найдется. Это подтверждают и его друзья, более того, они утверждают, что в этом плане его можно сравнить с рок-звездой.

Но, похоже, возраст все-таки берет свое, мальчик становится мужчиной. А, может быть, просто груз денег стал слишком велик, надо же их куда-то девать. Гейтс владеет 40 процентами акций Microsoft, и его личное состояние оценивается почти в 4 миллиарда долларов. "Да, у меня куча денег, — признает он, — и я совершенно свободен в том, что мне делать". Возможно, по этой причине он в последнее время довольно лихо взялся за дело. Сегодня Гейтс строит для себя в пригороде Сиэтла на берегу озера новый дом стоимостью 10 миллионов долларов. На территории 37000 квадратных футов будут располагаться бассейн, трамплин, кинотеатр, пляж, подземный гараж на 20 автомобилей, библиотека и столовая на 100 человек.

Еще одна новая слабость — это спортивные автомобили. Купленный им за 300 тысяч долларов Porsche все еще стоит в порту, так как таможня не пропускает его без справки о прохождении теста на безопасность водителя при дорожной аварии, требующейся по американским законам. Но дело в том, что эти модели слишком дорогостоящи, чтобы разбивать на стенде требуемые четыре машины. По этому поводу Гейтс и Аллен сегодня, как в добрые старые времена, на пару стряпают программу, имитирующую автокатастрофу, чтобы предъявить ее как гарантию безопасности автомобиля.

Б.Молчанов

По материалам:

D. Renzin, "Bill Gates — Soft Icon". Playboy, September, 1991.
Susann Lamers, "Programmers". Microsoft Press, 1986.
М.Кузмин, "Чудесная жизнь Иосифа Бальзамо, графа Калиостро. Петроград, 1919.

ВИКТОРИЯ

шлет пламенный ПРИВЕТ всем своим нынешним поклонникам!
говорит ДОБРО ПОЖАЛОВАТЬ своим будущим пользователям!
уже начала свое победное шествие по необъятным просторам одной шестой части суши.

Вы хотите добиться успеха, применив в своем бизнесе новые информационные технологии? И Вы думаете, что Вам удастся осуществить это без ВИКТОРИИ? А Вы знаете о тех могущественных возможностях ВИКТОРИИ, обладателем которых можете стать ВЫ? Неужели Вы в состоянии от всего этого отказаться?

ВИКТОРИЯ — это волшебная палочка в Ваших руках!

Вы программист? Вы пользователь? Вы новичок?
Новое компьютерное поколение выбирает ВИКТОРИЮ!
Не упустите счастливый случай!

Вам нравится Norton Commander? PCTools? XTree?
Да ведь Вы еще не работали с ВИКТОРИЕЙ!

ВИКТОРИЯ — это Ваша СИЛА
ВИКТОРИЯ — это Ваше МОГУЩЕСТВО
ВИКТОРИЯ — это Ваше ПРЕИМУЩЕСТВО
Ваш ИНТЕЛЛЕКТ и Ваша ВИКТОРИЯ сделают Вас НЕПОБЕДИМЫМ!

И Вы все еще сомневаетесь, какую оболочку Вам выбрать?

ВИКТОРИЯ доступна всем! Мы поддерживаем предельно низкие цены!

ВИКТОРИЯ — это атомная бомба в Вашем компьютере

ВИКТОРИЯ — это новая SOFT-БОМБА!

ВИКТОРИЯ — это ваша рабочая лошадка!

У Вас есть ВИКТОРИЯ! Ваши конкуренты в панике!

Работать с ВИКТОРИЕЙ — хороший тон для бизнесмена!

Вы приобрели ВИКТОРИЮ! Победа у Вас в кармане!

Phones: (095) 420-83-80
(095) 491-01-53

Сделай сам

Взять высушенное сердце жабы, залить отваром корней можжевельника, добавить пепел перьев черного петуха, двух скорпионов и зуб дракона и выпаривать после захода солнца до первого крика совы.

Старинный рецепт изготовления тонера для лазерного принтера.

В своей и без того нелегкой жизни советский программист сталкивается с массой мелких проблем, которые в своей нерешенности способны напрочь отравить радость общения с ЭВМ. Основная причина тому — полная неразвитость сферы текущего технического обслуживания. Конечно, кое-какую помощь получить можно, но стоит она безумных денег. Вот и приходится пользователям разбирать груды научно-технической литературы, брать в руки отвертки, тестеры, паяльники.

Чаще всего к заминкам в работе приводит истощение красящей ленты в принтере. Часть пользователей давно нашли способ легко выходить из этого затруднения, однако, как показывает жизнь, большинство идут к кооператорам и покупают ленточки по 30, а то и по 60 рублей за штуку. Когда мы несколько лет тому назад столкнулись с подобной проблемой, то решили ее раз и навсегда, и теперь хотим поделиться опытом с остальными пользователями. Заранее хочу сказать, технология очень проста и, вероятно, многие делают так, поэтому мы не претендуем на первопроходство и сверхоригинальность.

Начнем с самого начала...

Выбор ленты

Лента, которую можно поставить в принтер, должна обладать такими свойствами, как:

- высокая механическая прочность волокна. Под воздействием иголок принтера лента не должна давать ворс и мелкую пыль, которые могут забить печатающую головку;
- хорошая свариваемость. Нормальный, крепкий, аккуратный шов может дать только сварка, а не склейка или сшивание;
- эластичность. Волокна ленты должны быть достаточно эластичны, иначе отпечатанные точки будут иметь неправильную форму, а это очень некрасиво.

Всем этим требованиям отвечают ленты на полиамидной основе. Рассмотрим полиамидные ленты, которые может достать обычный пользователь.

1. Ленты производства бывшей ГДР, поставлявшиеся с принтерами ROBOTRON.

Ленты очень хорошие. Их отличает:

- высокое качество печати;
- высокая насыщенность и стойкость красящего слоя.

Надо отметить, что краска бывает двух оттенков: черная и фиолетовая.

У лент этого типа есть два недостатка — слишком жирная печать при свежей ленте (легко смазать напечатанное) и сложность в доставании.

2. Ленты производства Республики Польша, поставляемые с принтерами D100.

Ленты не очень высокого качества.

Их отличает:

- нестабильность характеристик. В одной партии встречаются и жесткие, и эластичные ленты, полувысохшие и свежие;
- ленты относительно быстро истощаются;
- высота ленты несколько выше стандартной (впрочем, и сам D100, мягко говоря, несколько нестандартен) и, поэтому, она с трудом транспортируется в EPSON-овском картридже. Для исправления этого недостатка приходится не полностью закрывать крышку. Лента поставляется сваренной в кольцо и в специальной коробке, которая, по идее, должна облегчать заправку в картридж. Однако эта коробка не подходит ни к EPSON-овским, ни к STAR-овским картриджам, так что толку от нее никакого.

3. Импортные ленты для пишущих машинок. Ленты хорошие. Мы встречали индийские и чехословацкие.
4. Лента для пишущих машинок производства СССР. На удивление неплохая лента, к тому же самая дешевая. Обладает высокой эластичностью, хорошей свариваемостью. Самый большой плюс — легче всего достать. Честно говоря, я предпочитаю именно ее и считаю, что в эксплуатации лучше только ГДР-овская, впрочем, это мое субъективное мнение). Для

тех, кто сразу побежал в магазин, привожу полное наименование этой ленты: "Лента для пишущих машин черная красящая текстильная полиамидная. ТУ 13-7309005 622-85" ширина 13 мм. Сейчас она стоит 1.80 руб., а до 2 апреля ее цена была 90 коп.

Теперь будем считать, что лентой мы запаслись, и пойдем дальше...

Сварка

Сварка ленты — главная технологическая операция. Она очень проста, но от того, как вы ею овладеете, будут зависеть быстрота и качество замены лент.

Прежде чем приступить к сварке лент, запаситесь необходимыми инструментами. Во-первых, выжигательным аппаратом. Если у вас его нет, то можно использовать нихромовую проволоку, закрепленную на ручке, и трансформатор с регулируемым напряжением. На худой конец подойдет хорошо разогретый паяльник с острым жалом. Во-вторых, вам понадобится ровная и гладкая деревянная дощечка. Она будет использоваться как подложка при сварке и не должна иметь лакокрасочного покрытия. Дело в том, что лак может легко воспламениться при касании раскаленной нихромовой проволокой. И, в-третьих, потребуется металлическая линейка. Линейка должна быть ровной, гладкой и достаточно увесистой. Хорошо подойдет стальная или медная полоска толщиной 2-3 мм.

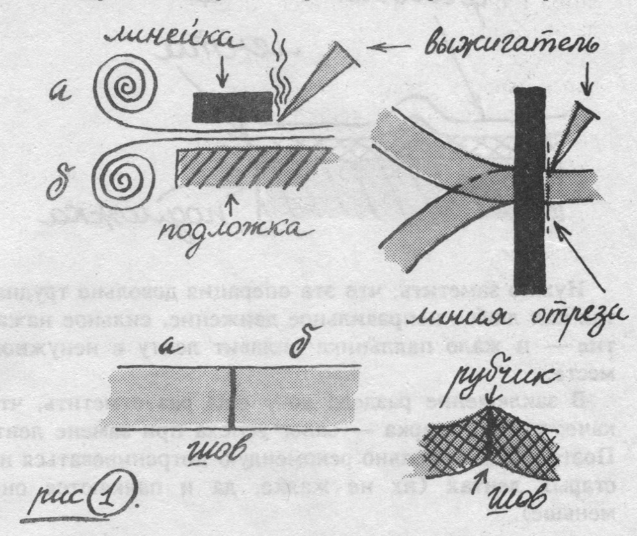
Сам процесс сварки происходит следующим образом:

1. Свариваемые ленты накладываются друг на друга свободными концами в одну сторону и кладутся на деревянную дощечку.
2. Вдоль места будущего шва прикладывается металлическая линейка, прижимающая ленты к дощечке. Шов желательно делать близко к свободным концам (примерно 1-2 см) — тогда будет меньше отходов.
3. Нагретым выжигателем отрезают свободные концы. При этом обе ленты расплавляются в месте разреза и слипаются.
4. Не отнимая линейки, отделяют отрезанные концы, после чего сваренные ленты можно извлечь.

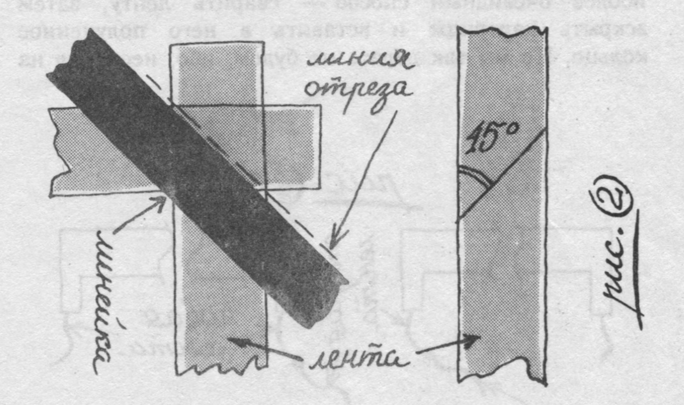
Для большей надежности сварного шва можно через 0,5-1 секунду после отрезки снять линейку и быстро наложить ее на еще мягкий шов и слегка прижать. При этом ленты склеиваются плотнее.

При сварке надо обращать внимание на следующие моменты. Отрезка должна производиться выжигателем, когда нихромовая проволока нагрета до малиново-красного цвета. При более низкой температуре (темный цвет) может получиться некачественный шов, а при более высокой (соломенно-желтый цвет) возможно воспламенение подложки или быстрое перегорание нихромовой проволоки выжигателя. Использование деревянной дощечки и металлической линейки себя полностью оправдывает. Деревянная подложка позволяет надежно, без проскальзывания, прижимать ленту, а, кроме того, запах от прожигаемой древесины намного приятнее, чем от текстолита или пластика. Металлическая линейка обеспечивает хороший отвод тепла от сварного шва (т.е. шов быстрее твердеет), достаточное сцепление с древесиной (надежная фиксация ленты) и удобство в работе. Постарайтесь, по возможности, не сваривать разнотипные ленты. Как правило, химический состав материалов основ разных лент хоть немного, да различается, а значит различаются и их температуры плавления. При этих условиях практически невозможно получить прочный шов. Единственный случай, когда это оправдано, будет описан ниже.

Весь процесс сварки подробно изображен на рис.1.

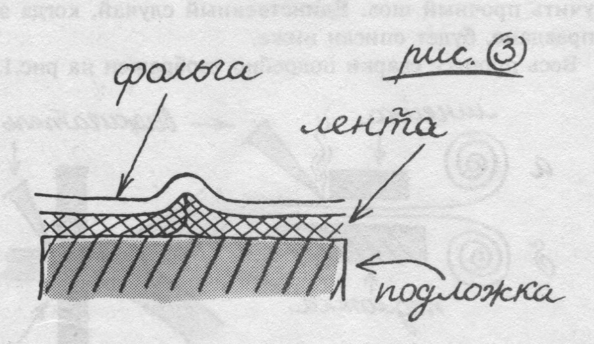


Описанным выше способом получают прямой шов. Если вы, по каким-то причинам, хотите получить косой шов, то свариваемые ленточки надо сложить крест-накрест и произвести отрезку по диагонали (см. рис.2).



Однако, как показывает опыт, косой шов не оправдывает себя, ибо краска в ленточке истощается раньше, чем успевает прорваться аккуратно выполненный прямой шов. Кроме того, изготовление косого шва требует хорошего глазомера и аккуратности, так как приложить линейку и сделать отрез надо строго по диагонали.

В ряде случаев вам придется загладить шов (ряд принтеров рвет его). Для этого лента расстилается на ровном месте рубчиком шва кверху и на нее накладывается тонкая алюминиевая фольга (конечно, лучше использовать тонкий фторопласт, но не все его могут найти). Для обозначения шва пригладьте получившийся "сэндвич", и тогда на фольге появится отпечаток. После этого несильно нагретым паяльником несколько раз прогладьте шов (рис.3).

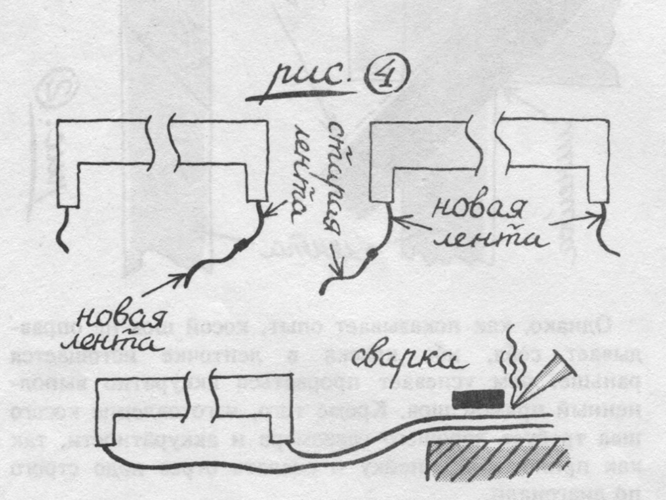


Нужно заметить, что эта операция довольно трудна, так как любое неправильное движение, сильное нажатие — и жало паяльника оплавит ленту в ненужном месте.

В заключение раздела хочу еще раз отметить, что качественная сварка — залог успеха при замене лент. Поэтому настоятельно рекомендую потренироваться на старых лентах (их не жалко, да и пачкаются они меньше).

Заправка картриджа

Если вы успешно овладели сваркой лент, то теперь самое главное — научиться заправлять картридж. Наиболее очевидный способ — сварить ленту, затем вскрыть картридж и вставить в него полученное кольцо. Но мы так делать не будем, ибо, несмотря на кажущуюся простоту, способ довольно трудоемок и грязен, а частое вскрытие картриджа отнюдь не благоприятно сказывается на сроке его службы.

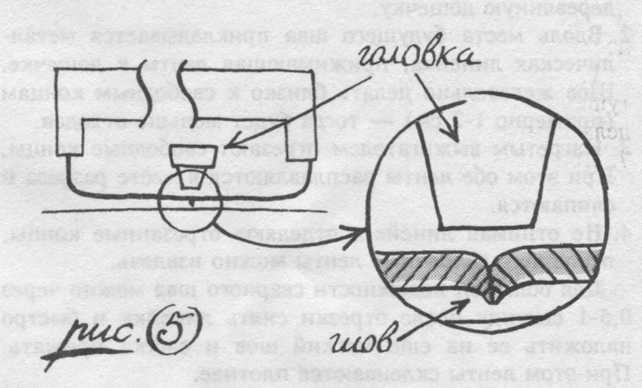


Способ заправки ленты (см. рис. 4).

1. Разрезается старая лента. Резать надо примерно посередине.
2. К концу, который при транспортировке втягивается в картридж, приваривается новая лента (это и есть тот случай, когда приходится сваривать разнотипные ленты).
3. Путем промотки свежая лента заправляется в картридж, а старая извлекается из него. При желании можно заправить несколько лент, сваривая одну с другой.
4. Полученные концы новой ленты свариваются друг с другом.

Налицо несколько преимуществ:
- нет необходимости вскрывать картридж;
- автоматически обеспечивается правильная заправка ленты, так как ее укладка получается путем штатного транспортирования;
- так как контакт с лентой происходит только при ее сварке, шансов испачкаться, как поросенок, у вас много меньше.

При заправке обратите внимание на следующее: во всех случаях нельзя допускать перекручивания лент (даже если лента должна быть в виде кольца Мебиуса). Рубчик шва должен быть на противоположной от печатающей головки стороне ленты. В этом случае иголки будут меньше разбивать шов. Правильно заправленная лента показана на рис. 5.



Так как на катушке за 1.80 руб. содержится всего 8 метров ленты, к тому же очень тонкой, у вас возникнет естественное желание заправить в картридж ленты с нескольких катушек. Однако, злоупотреблять этим не стоит, ибо при повышенной плотности набивки ленты увеличиваются усилия, необходимые для ее транспортировки, что приводит к повышенному износу принтера, зубчатых роликов картриджа и ленты. Ориентировочно в картридж от узкого принтера EPSON можно спокойно заправить ленту с двух катушек, но лучше поэкспериментируйте сами с теми лентами, которые у вас есть.

В заключение необходимо заметить, что требуется, как минимум, 2 картриджа для принтера. В один заправляется новая лента для печати документов, а в другой — более "забитая" лента для текущей черновой печати. При окончательном истощении черновая лента заменяется на новую, и картриджи меняются ролями.

Что делать, если у вас в принтере узкая лента

К сожалению, даже при западном уровне стандартизации в некоторых типах принтеров используются ленты, ширина которых отличается от ширины лент, выпускаемых в нашей стране. Если у вас такой принтер, то не стоит огорчаться — есть простой способ поправить положение.

Сразу замечу, речь пойдет о получении лент с шириной 7-8 мм. Так как обычная лента имеет ширину 13 или 16 мм, то сразу возникает желание разрезать ее вдоль на две ленты требуемой ширины. В самом начале мы хотели воспользоватся резаком для кинопленок, но нас остановило то, что полученная линия разреза — потенциальный источник пыли и ворсинок. Поэтому мы сделали маленькийленторезный станок. Конечно, он несколько сложнее инструментов для сварки лент, но затраты на его изготовление мгновенно окупятся при эксплуатации принтера с узкой лентой. Обратите внимание на тщательность изготовления станка, так как от этого будет полностью зависеть качество получаемой ленты. Эскиз станка приведен на рис. 6. Я нарочно не привожу чертеж, ибо конкретная реализация зависит от имеющихся заготовок и деталей, а приведенные рекомендации считаю достаточными для самостоятельного изготовления.

1. Подающая катушка. 2. Тормоз подающей катушки. 3. Направляющая. 4. Лентоприжим. 5. Разделительный флажок. 6. Нихромовая проволока. 7. Приемная катушка. 8, 9. Оси. 10. Привод.

Подготовка к резке

Катушка с разрезаемой лентой (1) надевается на ось (8). Для пропуска проволоки (6) и разделительного флажка (5) на ленте делается продольный надрез. Лента укладывается на направляющую (3), прижимается лентоприжимом (4) и закрепляется на приемной катушке (7). Резка ленты.

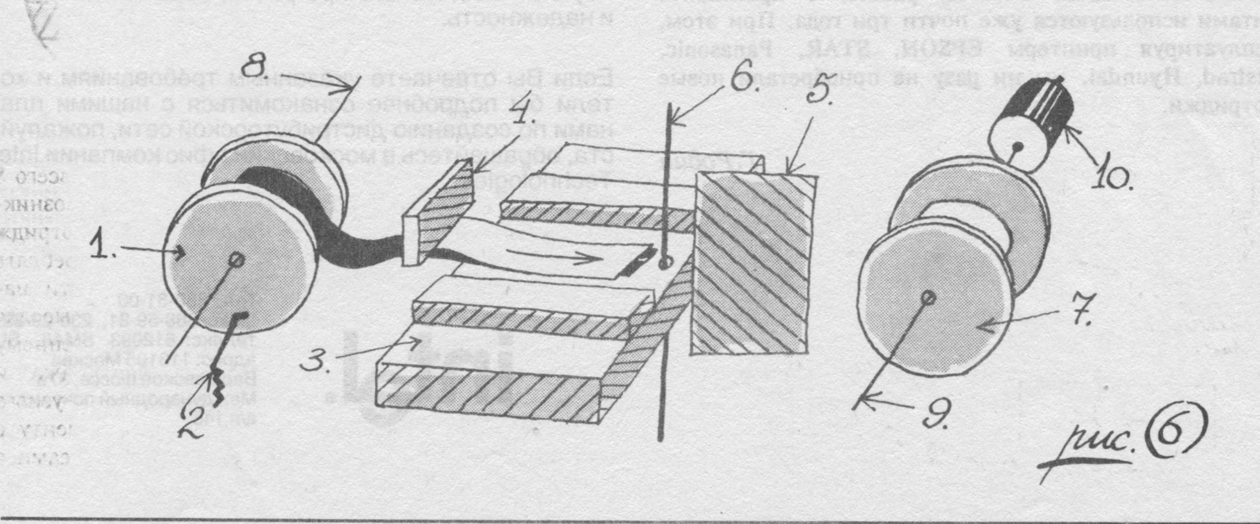
К проволоке (6) подводится напряжение, и вращением приемной катушки осуществляется транспортирование ленты. При этом раскаленная проволока разрезает ленту на две, ширина которых зависит от положения проволоки относительно направляющих. Температура нагрева проволоки и скорость транспортирования подбираются опытным путем. Очевидно, что эти параметры непосредственно влияют на качество резки и срок службы нихромовой проволоки, которая подвергается тепловому воздействию (значительно снижающему ее прочность и зависящему от подаваемого напряжения) и механическому, разрывному воздействию со стороны разрезаемой ленты (определяемому скоростью резки). Некоторые замечания по поводу назначения деталей станка.

1. Тормоз (2) служит для получения начального натяжения ленты.

2. Лентоприжим (4) и направляющая (3) обеспечивают правильное положение ленты при резке.

3. Нихромовая проволока в нагретом состоянии обеспечивает резку ленты. При этом происходит хорошая заделка линии отреза оплавлением и гарантируется отсутствие ворса и пыли при эксплуатации ленты.

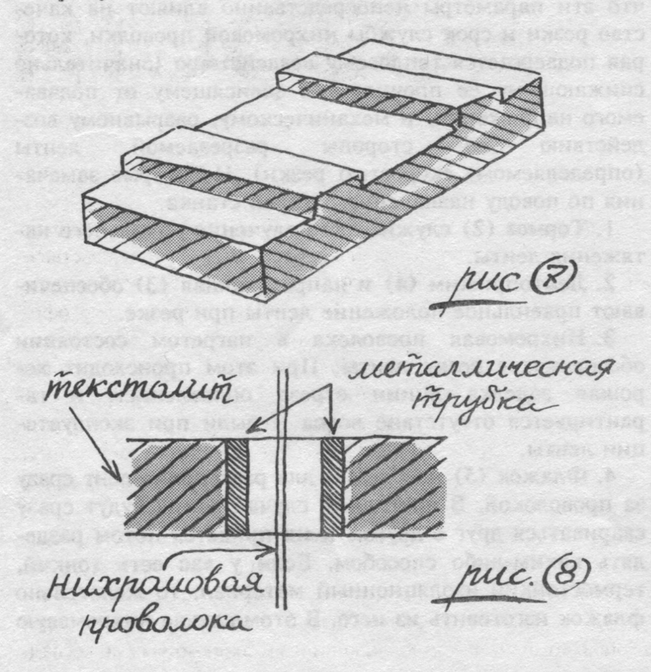
4. Флажок (5) необходим для разделения лент сразу за проволокой. В противном случае ленты будут сразу свариваться друг с другом и их придется потом разделять каким-либо способом. Если у вас есть тонкий, термостойкий изоляционный материал, то желательно флажок изготовить из него. В этом случае нихромовую



проволоку можно будет пустить по краю флажка, который будет воспринимать механические нагрузки от движущейся ленты. Срок службы проволоки значительно повысится.

5. Механический привод. Вообще можно обойтись одним воротком на приемной оси, но тогда будет трудно обеспечить равномерное движение ленты, что является залогом качественной резки. Если у вас есть возможность, то лучше использовать электродвигатель с редуктором.

В нашем случае направляющая сделана из текстолита с металлическими накладками (см. рис. 7), а отверстие под проволоку армировано тонкой стальной трубкой (отрезок иголки от шприца) (см. рис. 8). Такое армирование не позволяет раскаленному нихрому жечь текстолит.



Описанные выше способы работы с красящими лентами используются уже почти три года. При этом, эксплуатируя принтеры EPSON, STAR, Panasonic, Amstrad, Hyundai, мы ни разу не приобретали новые картриджи.

Г.Родин



НОВОСТИ

Банковское обслуживание через спутники началось в Индии 2 декабря, когда 26 банков этой страны стали абонентами сети Society for World-wide Interbank Financial Telecommunication (SWIFT). Среди участников State Bank of India, Bank of India, Bank of Baroda и Canara Bank, сейчас их терминалы подсоединены к центру SWIFT в Бомбее, откуда сообщения передаются в Лондон или Сингапур и далее в другие банки. Идея родилась в Индийской Банковской Ассоциации еще в 1982 году, но разрешение правительства было получено только в июне 1991 года. В деятельности SWIFT намечается новый этап — как раз тогда, когда начались испытания SWIFT II.

The Teleputing Hotline, December 10 1991

AT&T BELL LABS анонсировали "LuckyNet," схему работы сети со скоростью 2.5 Гбит/с, которая может функционировать с волоконно-оптическими или микроволновыми передатчиками.

The Teleputing Hotline, December 10 1991

Каждая индустрия проходит через этапы, которые легко опознать. На первом этапе пионеры испытывают трудности первопоселенцев, доказывая, что рынок все-таки существует. На втором этапе появляется толпа и начинается настоящая конкуренция. Третья стадия — перетряска. Она отмечается переходом в зрелое состояние и снижением конкуренции. Четвертый этап — зрелая индустрия — наиболее доходен.

Компьютеризация "в полевых условиях" (вне офисов) сейчас входит во вторую стадию. Образуется толпа конкурентов. Что касается беспроводных сетей, то за последние несколько недель BellSouth приобрела половину сети RAM Mobile, Global Star объявила о создании сети низкоорбитальных спутников, которая будет конкурировать с проектом Iridium компании Motorola, а McCaw Cellular и Oracle Systems подыскивают частоту для службы радиопередачи данных.

В каждой области компьютеризации вне офисов есть несколько конкурентов. Если вы предпочитаете радиочастоты, то к вашим услугам RAM и ARDIS. Если вам нравятся более высокие частоты, то — Mtel и MCaw. Производители модемов стремглав бросились эксплуатировать сотовые частоты, и планируются две спутниковые сети. Если взглянуть на терминалы, то программное обеспечение предлагают Go, Microsoft и Momenta, среди производителей нового аппаратного обеспечения такие гиганты, как IBM и AT&T. Без сомнения, основные японские производители не заставят себя долго ждать.

Все это означает, что вам нужно тщательно рассмотреть ваши собственные действия. Определите, какая часть ваших сотрудников работает вне офисов. Подумайте о том, как компьютерные ресурсы могут увеличить их производительность. Определите возможные решения. Выясните намерения конкурентов. И приготовьтесь к тому, чтобы купить оборудование в 1992 году.

Если вы этого не сделаете, это сделает ваш конкурент. В результате он предложит лучшее обслуживание или более низкие цены. А учитывая продолжающийся экономический спад, это и есть разница между скромными доходами и банкротством.

The Teleputing Hotline, December 5, 1991

Hayes Microcomputer Products и GPT открыли в Ноттингеме лабораторию разработки прикладных программ для ISDN (интегрированных цифровых сетей). Обе компании производят изделия для ISDN: Hayes выпускает адаптеры для компьютеров, GPT — телефоны ISDN. Две компании хотят, чтобы разработчики программного обеспечения использовали набор AT-команд фирмы Hayes и BIOS ISDN для создания реальных прикладных программ для предприятий в Европе и США. British Telecom сообщает, что

число абонентов ее службы ISDN 2 составляет десятки тысяч. ISDN 2 обеспечивает по двухпроводному телефонному соединению два канала данных со скоростью передачи 64,000 бит в секунду плюс управляющий канал. Обещается увеличение сети.

The Teleputing Hotline, December 5, 1991

Японское агентство Information Processing Agency сообщило, что за последние два месяца зарегистрировано рекордное число компьютерных вирусов. Ущерб, нанесенный ими, чрезвычайно невелик. Из 11 вирусов, о которых сообщалось в сентябре, восемь действовали на компьютере NEC PC-9801, являющимся стандартным в Японии. Большинство сообщений пришли из Токио. Новейший вирус называется "Vaccina" — это безвредное заболевание, инфицирующее машины с MS-DOS.

В США, по сообщению Dataquest, 63% владельцев персоналок столкнулись с вирусами, а 9% потерпели ущерб на двадцати пяти или более компьютерах. Самые серьезные опасности подстерегают большие компании. В США сейчас известно о примерно 1000 различных вирусных программах.

The Teleputing Hotline, December 5, 1991

CHARLES SCHWAB отменила взимание ежемесячной платы с клиентов, читающих котировки акций в системе GEnie в реальном времени. Поминутная оплата остается в силе.

GO CORPORATION откроет в Японии предприятие для разработки и сбыта японской версии своего программного обеспечения PenPoint для компьютеров с рукописным вводом в начале 1993 года.

HAYES, крупнейший производитель модемов для персональных компьютеров, сообщила что в 1991 году объем ее продаж в Европе вырос на 70%, в Азии — на 100%, в то время как общие доходы возросли на 15%.

MICROSOFT анонсировала шлюзы между своей системой Microsoft Mail и IBM Profs, факсом, X.400, SMTP/Unix, Novell MHS и MCI Mail.

В АВСТРАЛИЙСКОЙ ПРОВИНЦИИ Новый Южный Уэльс начала работу The Employment Network (TEN), он-лайновая служба, соединяющая нанимателей с потенциальными работниками.

REUTERS представила систему Decision 2000, которая дает возможность валютным биржевикам анализировать он-лайновую информацию агентства по облигациям перед сделками.

SHAREWARE PUBLISHING будет продавать пакет связи Odyssey PC в Великобритании. Компания прошлым летом отказалась от Procomm.

TELEBIT усовершенствует свой новый модем V.32bis, T3000, с помощью своего собственного протокола.

The Teleputing Hotline, December 5, 1991

Датская телефонная компания объявила об установлении волоконно-оптической и спутниковой связи с Россией. Используется спутниковая система американской компании Hughes, дающая возможность прямой телефонной связи между Восточной и Западной Европой. Сейчас, звоня по телефону, часто слышишь гудки "занято", сообщения типа "ждите ответа" или сталкиваешься с отсутствием сигнала на линии. С введением новой линии жить станет легче.

Подсоединение будет осуществлено через систему Eutelsat. Оптико-волоконная линия от Копенгагена до Кингисеппа будет создана датскими GN Store Nord A/S и Tele Danmark A/S. Уже заабонировано 15,360 каналов. Между Москвой и Кингисеппом придется устанавливать радиосвязь, так как Запад отказывается поставлять российскому правительству оптико-волоконный кабель. Система начнет работать весной 1993 года.

The Teleputing Hotline, December 3, 1991

Sierra Network присоединилась к Prodigy и Compuserve и стала третьей американской он-лайновой системой, предлагающей в розничной торговле наборы для начала работы в своей системе. Sierra является игровой системой. Start-Up Kit стоит 29.95 долл. Предлагается набор: шахматы, электронный корректор, бридж, минигольф, он-лайновые беседы. Приложения включают электронную почту, конференции и путеводители по играм Sierra.

The Teleputing Hotline, December 3, 1991

Ericsson GE подписала соглашение с Anterior Technology и Research In Mo-

tion (RIM) о разработке беспроводного шлюза для электронной почты. Шлюз Radiomail фирмы Anterior сможет соединит сеть RAM Mobile Data Mobitex с такими системами, как ATT Mail, почтовыми системами ЛВС типа Lotus cc:Mail, а также с сетями TCP/IP Internet и UUCP/USENET. RIM создаст программное обеспечение, дающее портативным компьютерам доступ к шлюзу через радиомодемы Ericsson GE. Первым устройством, имеющим такие возможности, станет palmtop-компьютер HP 95LX.

The Teleputing Hotline, December 3, 1991

Askri анонсировала сделанный в Советском Союзе пакет шифрации для IBM PC, отвечающий Data Encyption Standard (Стандарт шифрации данных). Cryptos является первым пакетом такого рода в России. Это стандарт, использующийся Агентством Национальной Безопасности США, которое сейчас пытается ввести другой алгоритм. Американские компании возражают против этого, так как они вложили деньги в DES, а также потому, что правительство получит возможности по шифрации, которых не будет у частных фирм. Кирилл Чащин пишет для Newsbytes, что главное для советских пользователей то, что их компьютеры могут теперь работать с DES, даже если создание новых ключей на IBM PC AT занимает час. Стоить Cryptos будет 5,000 рублей.

The Teleputing Hotline, November 26, 1991

APPLE выходит на рынок мобильных устройств приема-передачи данных с помощью интерфейса сотовых модемов Axsys фирмы Spectrum Cellular.

BRITISH TELECOM приобрела компанию FTT Alphanumeric, которая производит системы программного обеспечения для заключения сделок, используемые брокерами на биржах.

NTT выпустила изделие, называемое ею самым надежным факсимильным аппаратом в мире, который снабжен мощными возможностями по шифрации и дешифрации. Модели серии T используют криптографическую систему FEAL-8 NTT.

The Teleputing Hotline, November 26, 1991

Ограничения на торговлю с Чехословакией, Венгрией и Польшей, основанные на соображениях безопасности, могут быть скоро сняты, так что эти страны смогут купить волоконно-оптические телефонные системы. Россия по-прежнему не может этого сделать, в то время, как Китай и Иран продолжают развивать свои надежные волоконные сети. В любом случае из списка высокотехнологичных товаров, запрещенных к экспорту, вычеркивается все больше и больше изделий, включая компьютеры типа IBM PC. В COCOM входят все члены военного союза НАТО, кроме Исландии, плюс Австралия и Япония.

The Teleputing Hotline, November 21, 1991

APPLE направила запрос правительству о введении 63% тарифа на ввоз активных жидкокристаллических матричных плоских дисплеев, типа того, который используется в портативном Macintosh´е. Многие американские производители компьютеров-блокнотов уже перенесли производство в другие страны, так как они в таких условиях не могут создать конкурентоспособные продукты дешевле 20,000 долларов, против 2,000 долларов за японские машины.

HEWLETT-PACKARD и сингапурские фирмы представили факс-аппарат серии Group IV, использующий линии ISDN и работающий даже когда пользователи по той же самой линии ведут обычный разговор.

NEC, Matsushita Electric и Mitsubishi Electric готовят соглашение по совместной разработке интегральных схем для телевидения высокой четкости. К союзу могут также присоединиться AT&T Microelectronics и LSI Logic.

The Teleputing Hotline, November 21, 1991

"БАЗЫ ДАННЫХ И ЭКСПЕРТНЫЕ СИСТЕМЫ"

От кого ______________________________

Адрес ______________________________

____

На (шесть, три) ______________ выпусков. Количество экз. ______________

Прошу оформить подписку на 1992 год

Подписная плата в сумме ______________________ руб. перечислена

платежным поручением No. ____________ от ________________ 199__ г.

(Копия платежного поручения прилагается)

Руководитель ______________________________

(подпись, фамилия, инициалы)

Примечание. Заказ высылать по адресу: 123459 Москва а/я 20

Заказ

Советско-американское предприятие "Соваминко"
Рекламно-издательское агентство "КомпьютерПресс"

Принимает заказы на журнал "КомпьютерПресс" и производит отправку наложенным платежом.

Заказ высылается по адресу: 191186, Санкт-Петербург, Невский проспект, 28
Магазин № 1 "Дом Книги"

От кого ....

Адрес ....
(почтовый индекс указывать обязательно)

....

Номера выпусков .... **Количество экземпляров** ....

Заказ

Советско-американское предприятие "Соваминко"
Рекламно-издательское агентство "КомпьютерПресс"

Принимает заказы на журнал "КомпьютерПресс" и производит отправку наложенным плетжом

Заказ высылается по адресу: 630076, Новосибирск, Красный проспект, 60
Магазин № 7 "Техническая книга"

Телефон для справок 20-05-09

От кого ....

Адрес ....

....

Номера выпусков .... **Количество экземпляров** ....